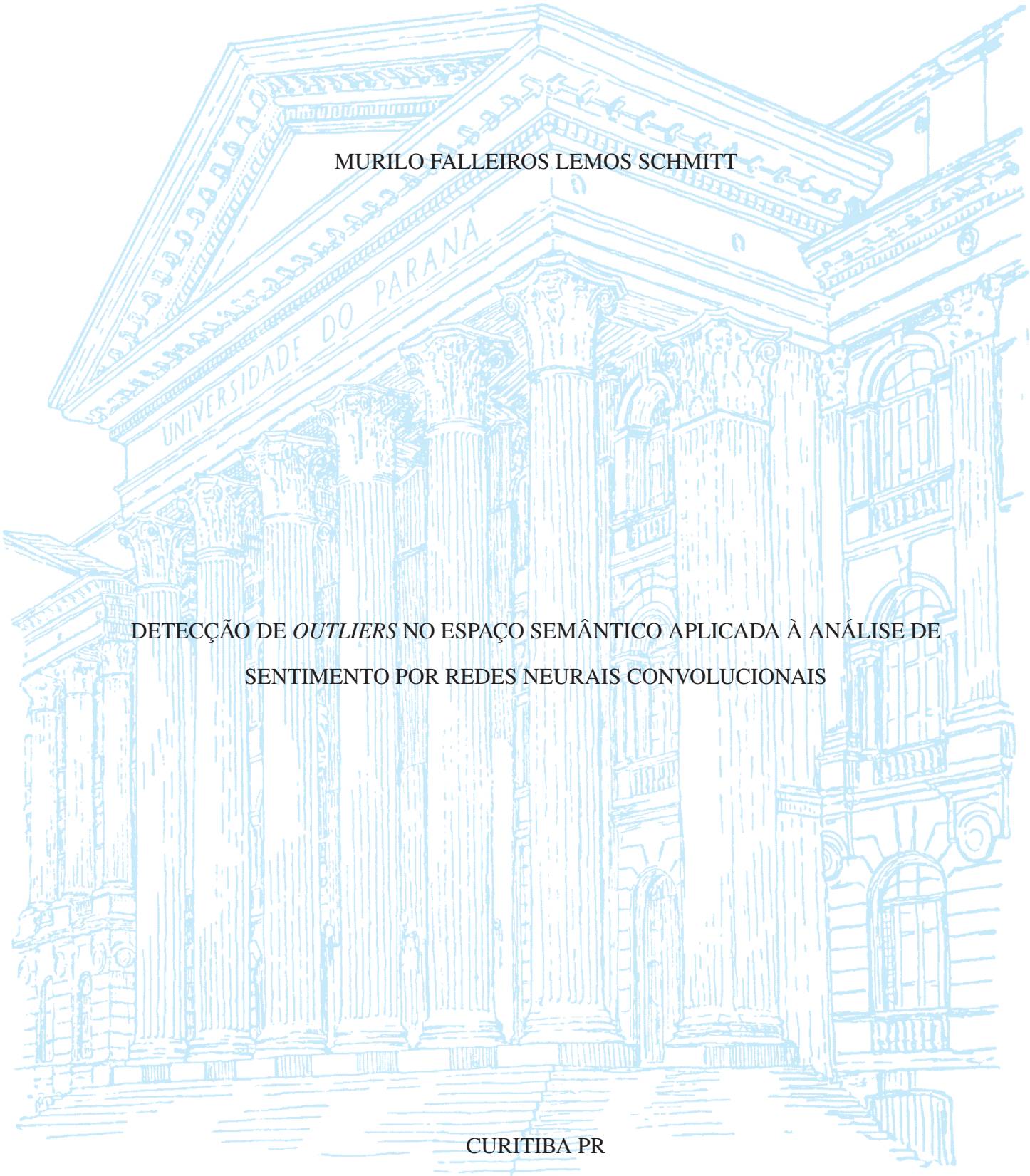
UNIVERSIDADE FEDERAL DO PARANÁ

MURILO FALLEIROS LEMOS SCHMITT



DETECÇÃO DE *OUTLIERS* NO ESPAÇO SEMÂNTICO APLICADA À ANÁLISE DE SENTIMENTO POR REDES NEURAIS CONVOLUCIONAIS

CURITIBA PR

2018

MURILO FALLEIROS LEMOS SCHMITT

DETECÇÃO DE *OUTLIERS* NO ESPAÇO SEMÂNTICO APLICADA À ANÁLISE DE SENTIMENTO POR REDES NEURAIS CONVOLUCIONAIS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Eduardo Jaques Spinosa.

CURITIBA PR

2018

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

S355d Schmitt , Murilo Falleiros Lemos
Detecção de Outliers no Espaço Semântico Aplicada à Análise de Sentimento por Redes Neurais Convolucionais / Murilo Falleiros Lemos Schmitt . – Curitiba, 2018.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2018.

Orientador: Eduardo Jaques Spinosa . -

1. Redes Neurais (computação). 2. Processamento de linguagem natural (Computação) 3. Análise de Sentimento. I. Universidade Federal do Paraná. II. Spinosa, Eduardo Jaques . III. Título.

CDD: 006.3

Bibliotecária: Vanusa Maciel - CRB - 9/1928

MINISTÉRIO DA EDUCAÇÃO
SETOR SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **MURILO FALLEIROS LEMOS SCHMITT** intitulada: **Detecção de Outliers no Espaço Semântico Aplicada à Análise de Sentimento por Redes Neurais Convolucionais**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 22 de Agosto de 2018.

EDUARDO JAQUES SPINOSA
Presidente da Banca Examinadora (UFPR)

MYRIAM REGATTIERI DE BIASE DA SILVA DELGADO
Avaliador Externo (UTFPR)

EDUARDO TODT
Avaliador Interno (UFPR)

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

RESUMO

Análise de sentimento é uma tarefa importante na área de Processamento de Linguagem Natural que consiste em automaticamente atribuir documentos de texto a classes previamente definidas que representam sentimentos ou opiniões positivas/negativas em relação a um determinado assunto. Para resolução dessa tarefa, podem ser utilizadas técnicas de aprendizado de máquina. No entanto, para que possam atingir uma boa capacidade de generalização, essas técnicas dependem de um pré-processamento cuidadoso e de uma representação adequada dos dados. Este trabalho propõe tratar essas questões fundamentais por meio de redes neurais convolucionais e algoritmos de agrupamento baseados em densidade. As representações de palavras utilizadas neste trabalho foram obtidas de vetores previamente treinados de maneira não-supervisionada, denominados *word embeddings*. Essas representações são capazes de capturar informações sintáticas e semânticas das palavras, o que leva palavras similares a serem projetadas próximas no espaço semântico. Neste cenário, o modelo proposto utiliza um algoritmo de agrupamento no espaço semântico para extrair informações adicionais das representações vetoriais das palavras com o objetivo de melhorar o desempenho da rede neural convolucional. Utilizou-se um algoritmo de agrupamento baseado em densidade para detecção e remoção de *outliers* dos documentos a serem classificados, antes desses documentos serem treinados e classificados pela rede neural convolucional. Para análise do modelo proposto, foram conduzidos experimentos com dois algoritmos de obtenção de *word embeddings* sobre cinco bases de dados, estudando-se o impacto da remoção de *outliers* em diferentes graus de intensidade. Os resultados demonstram que os *outliers* têm pouco impacto na taxa de acerto do classificador, podendo aumentar ligeiramente, mas sua remoção pode impactar positivamente no desempenho em termos de tempo de execução da rede.

Palavras-chave: *Deep Learning*. Detecção de *Outliers*. Redes Neurais Convolucionais. Análise de Sentimento.

ABSTRACT

Sentiment analysis is an important task in Natural Language Processing that consists in automatically assigning text documents to predefined classes that represent sentiments or a positive/negative opinion about a subject. To solve this task, machine learning techniques can be used. However, in order to achieve good generalization, these techniques require a thorough preprocessing and an appropriate data representation. To deal with these fundamental issues, this work proposes the use of convolutional neural networks and density-based clustering algorithms. The word representations used in this work were obtained from vectors previously trained in an unsupervised way, denominated word embeddings. These representations are able to capture syntactic and semantic information of words, which leads to similar words being projected closer together in the semantic space. In this scenario, in order to improve the performance of the convolutional neural network, the use of a clustering algorithm in the semantic space to extract additional information from the data is proposed. A density-based clustering algorithm was used to detect and remove outliers from the documents to be classified before these documents were used to train the convolutional neural network. To evaluate the proposed method, experiments were conducted with two different embeddings across five datasets, by studying the impact of the removal of outliers in different degrees of intensity. Results show that the outliers have little impact on the classifier's accuracy, being able to improve it slightly, but their removal can have positive impact on performance in terms of network runtime.

Keywords: Deep Learning. Outlier Detection. Convolutional Neural Networks. Sentiment Analysis.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

CBOW	*Continuous bag-of-words*
CharSCNN	*Character to Sentence Convolutional Neural Network*
CNN	*Convolutional Neural Network*
DBSCAN	*Density-based Spatial Clustering of Applications with Noise*
DNN	*Deep Neural Network*
DCNN	*Dynamic Convolutional Neural Network*
gloVe	*Global Vectors*
GPU	*Graphics Processing Unit*
MLP	*Multilayer Perceptron*
NLP	*Natural Language Processing*
NLTK	*Natural Language Toolkit*
RNA	Redes Neurais Artificiais
SGD	*Stochastic Gradient Descent*

LISTA DE SÍMBOLOS

α Limiar de nível de significância

δ Distância entre pontos de maior densidade

ρ Densidade local do ponto

SUMÁRIO

1 INTRODUÇÃO

Classificação de texto é um problema importante da área de Processamento de Linguagem Natural (NLP, *Natural Language Processing*). Classificação de texto consiste em atribuir documentos de texto de diferentes tópicos em categorias pré-definidas. Classificação de texto pode ser usada para detectar e classificar documentos em um conjunto de tópicos, detecção de *spam*, avaliar uma análise de um produto e análise de sentimento (Manning e Schutze, 1999) (Liddy, 2001) (Liu, 2012) (Johnson e Zhang, 2015a).

Análise de sentimento é uma área de pesquisa que analisa as opiniões, sentimentos e avaliações de pessoas em relação a diferentes produtos, serviços, organizações ou tópicos. É uma área de pesquisa que possui impacto em diversos domínios de aplicação como serviços financeiros, eventos sociais, eleições políticas, marketing e serviços a clientes, entre outros. Por conta da facilidade de acesso à informação disponibilizada pela Internet, essa área de pesquisa ganhou renovado interesse e atualmente é bastante ativa e importante. Problemas de análise de sentimento podem ser formulados como problemas de classificação de texto, tal que cada documento pode ser classificado em classes pré-determinadas, como positivo e negativo (Pang e Lee, 2008) (Liu, 2012).

Modelos de aprendizado de máquina podem ser utilizados para os problemas de classificação de texto e análise de sentimento. Esses modelos consistem em representar a entrada (documento) como um vetor e passar esse vetor para um classificador que irá atribuir uma classe ao documento. O bom desempenho de um classificador para a tarefa de classificação de texto normalmente depende de pré-processamento e *feature engineering* (n-gramas, contagem de palavras chave, remoção de *stop words*) (Collobert et al., 2011) (Liu, 2012) (Goldberg, 2016).

A escolha de características para representar a frase é um processo empírico, e a escolha dessas características depende do domínio da aplicação. Dessa maneira, são exigidos novos experimentos e pesquisas para novos domínios. Uma alternativa para esse problema é utilizar aprendizado de máquina para descobrir não somente como mapear a representação para a saída, mas também a própria representação: essa abordagem é conhecida como *representation learning* (aprendizagem de representação) (Collobert et al., 2011) (Goodfellow et al., 2016).

1.1 *DEEP LEARNING*

Técnicas convencionais de aprendizado de máquina possuem limitações em processar dados em forma bruta. Essas técnicas exigem conhecimento profundo e específico do problema para obter um extrator de características que transforme esses dados brutos em informações que serão compreendidas pelo classificador para que o mesmo atribua os dados a uma classe (LeCun et al., 2015).

Deep Learning utiliza uma abordagem conhecida como aprendizagem de representação. Aprendizagem de representação é um conjunto de métodos que permite que o sistema trabalhe diretamente com os dados brutos e automaticamente descubra a representação necessária para a classificação. Esse tipo de abordagem em geral obtém bons resultados com uma melhor capacidade de generalização (LeCun et al., 2015) (Goodfellow et al., 2016). *Deep Learning*

são métodos com múltiplos níveis de representação, obtidos pela composição de módulos não-lineares, tal que cada módulo transforma a representação de um nível para uma representação de nível mais abstrato. Com a composição dessas transformações, funções complexas podem ser aprendidas. Essas camadas de extração de características não são obtidas por influência humana, mas sim por uma sequência de passos de aprendizagem aplicados aos dados puros (LeCun et al., 2015).

1.2 *CONVOLUTIONAL NEURAL NETWORKS*

Convolutional Neural Networks (CNNs) são um modelo de *deep learning* que vem sendo utilizado com sucesso há algum tempo (LeCun et al., 1998) e, com o rápido processamento oferecido pelas placas gráficas (GPU, *Graphics Processing Unit*), CNNs têm se tornado essenciais na obtenção de ótimos resultados em visão computacional. CNNs são redes neurais, mas que aplicam convolução ao invés de multiplicação de matrizes em pelo menos uma camada (Krizhevsky et al., 2012) (Schmidhuber, 2015) (Goodfellow et al., 2016).

CNNs são projetadas para processar dados em forma de matrizes, e são compostas basicamente de duas operações principais: convolução e *pooling*. CNNs possuem propriedades interessantes, pois, além de descobrir características da entrada de forma automática por meio de treinamento, são capazes de aprender representações hierárquicas dos dados. Essas representações contêm propriedades importantes, como composição (recursos de alto nível são obtidos por meio da composição de níveis mais baixos), e alto grau de invariância para translação, variação de dimensão e distorções. Nas imagens, *pixels* próximos são altamente correlacionados e invariantes à posição na imagem: isso permite a detecção de grupos de *pixels* em diversas posições da imagem. Essas propriedades são encontradas também em textos (Haykin, 1999) (LeCun et al., 2015), o que torna as CNNs atrativas para a aplicação aos problemas de classificação de texto e análise de sentimento, e têm sido utilizadas recentemente, obtendo ótimos resultados (Kalchbrenner et al., 2014) (Kim, 2014) (Johnson e Zhang, 2015a).

1.3 REPRESENTAÇÃO DE DADOS

Diferentemente de outras áreas, NLP possui um problema fundamental: textos não podem ser interpretados diretamente por um classificador. Dessa maneira, é necessária uma forma de mapear os dados de forma que possam ser aplicados no classificador (Sebastiani, 2002). Normalmente, cada palavra do texto é associada a um vetor, e a sequência de vetores é utilizada como entrada para o classificador. Usualmente, duas formas são utilizadas para essa representação: *one-hot*, tal que cada palavra presente no texto é associada a um vetor, cada vetor possui tamanho igual ao tamanho do vocabulário, e um elemento do vetor indica qual é a palavra (Johnson e Zhang, 2015a). Essa representação apresenta problemas na dimensionalidade dos vetores, pois em casos em que o tamanho do vocabulário é muito grande, cada palavra será representada por um vetor de tamanho igual ao tamanho do vocabulário. A segunda forma de representação são métodos de frequência, sendo o mais tradicional a representação *bag-of-words*, tal que cada documento é representado por um vetor do tamanho do vocabulário, contendo o número de ocorrências das palavras no texto, sendo cada índice do vetor uma palavra do vocabulário (Pang e Lee, 2008) (Johnson e Zhang, 2015a). Esse método possui como limitações a não preservação da ordem das palavras nos documentos, o que torna esse método inadequado para análise de sentimento.

Recentemente, uma nova forma de se representar palavras foi proposta: *word embeddings*. *Word embeddings* consistem em uma representação obtida após criar um mapeamento que projeta

palavras em um espaço vetorial de menor dimensão. Esse tipo de representação é capaz de capturar relações semânticas entre as palavras, além de solucionar o problema de dimensionalidade (Mikolov et al., 2013a) (Pennington et al., 2014).

1.4 *CLUSTERING*

Word embeddings são capazes de representar muitas regularidades e padrões linguísticos, e, muitos desses padrões podem ser obtidos por operações lineares, como operações simples com vetores (Mikolov et al., 2013b). Essas características sugerem que palavras com semânticas parecidas são projetadas próximas em dimensões menores, o que torna interessante a ideia de agrupar essas palavras. Para esse propósito, algoritmos de agrupamento (*clustering*) podem ser utilizados. Algoritmos de agrupamento consistem em particionar um conjunto de objetos em grupos (*clusters*). Baseado em uma determinada descrição dos dados, o objetivo de um algoritmo de agrupamento é colocar objetos similares em um mesmo grupo, e objetos não similares em grupos diferentes (Manning e Schutze, 1999) (Duda et al., 2001). A aplicação de algoritmos de agrupamento permite capturar informações relevantes sobre uma distribuição de dados, como por exemplo a detecção de *outliers*, que consiste em identificar elementos cujas características diferem fortemente dos demais (He et al., 2003).

1.5 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é o estudo da detecção de *outliers* no espaço semântico no contexto do problema de análise de sentimento, obtidos por meio da aplicação de um algoritmo de agrupamento baseado em densidade sobre as representações vetoriais de palavras (*word embeddings*). Os *outliers* são detectados para serem removidos dos vocabulários antes do treinamento e classificação dos documentos por meio de uma CNN.

A motivação para tal remoção vem das características do problema de análise de sentimento, em que a polaridade de cada documento pode ser caracterizada por um subconjunto de palavras-chave. O objetivo é, portanto, remover palavras que tenham pouco impacto na caracterização do sentimento associado a cada documento. No âmbito deste trabalho, essas palavras são consideradas *outliers* por pertencerem a regiões de baixa densidade, ou seja, regiões que contêm palavras que não são similares a outras palavras no espaço semântico. A remoção de palavras busca melhorar o desempenho de CNNs em termos de taxa de acerto e tempo de execução.

1.6 ORGANIZAÇÃO DO TRABALHO

O trabalho estrutura-se da seguinte maneira:

- **Capítulo 2** introduz os principais conceitos aplicados nessa dissertação, incluindo Processamento de Linguagem Natural, *Deep Learning* e *Convolutional Neural Networks* e algoritmos de agrupamento e detecção de *outliers*.
- **Capítulo 3** apresenta os principais trabalhos relacionados da área de CNNs para classificação de textos.
- **Capítulo 4** apresenta a proposta deste trabalho, que é um modelo para o problema de análise de sentimento, baseado na detecção de *outliers* no espaço semântico por meio de um algoritmo de agrupamento baseado em densidade, com a classificação realizada por uma CNN.

- **Capítulo 5** descreve a etapa experimental realizada, incluindo a metodologia adotada, os resultados obtidos e discussões sobre esses resultados.
- **Capítulo 6** apresenta as considerações finais deste trabalho e propostas para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os conceitos necessários para a compreensão dessa dissertação. Inicialmente são introduzidos conceitos de Processamento de Linguagem Natural, incluindo análise de sentimento e representação vetorial de palavras. Em seguida, são apresentados embasamentos de *Deep Learning* e Redes Neurais Convolucionais (CNNs). Depois, são contextualizados algoritmos de agrupamento, incluindo algoritmos de agrupamento baseados em densidade. Por fim são apresentados conceitos de detecção de *outliers*.

2.1 PROCESSAMENTO DE LINGUAGEM NATURAL

Processamento de Linguagem Natural (NLP) é uma área de pesquisa que explora como computadores podem compreender e manipular textos escritos em linguagem natural a fim de automatizar processos e realizar tarefas importantes (Chowdhury, 2003).

A área de NLP busca converter a linguagem humana em uma representação que seja fácil para a compreensão de computadores, consistindo de técnicas computacionais para analisar e representar textos, com o intuito de permitir o mesmo processamento de linguagem realizada por humanos (Liddy, 2001) (Collobert e Weston, 2008).

Pesquisa na área de NLP visa reunir conhecimentos sobre como os seres humanos entendem e usam linguagem para desenvolver ferramentas e técnicas adequadas para que sistemas computacionais compreendam e manipulem linguagens naturais para a realização de tarefas desejadas, tais como: recuperação e extração de informação, sumarização de textos, análise de documentos, tradução automática e classificação de texto (Chowdhury, 2003) (Collobert e Weston, 2008).

2.1.1 Classificação de Texto

Classificação de texto (ou categorização de texto) é a tarefa de atribuir um objeto (texto) a uma classe pré-definida, como um tema, ou sentimento. O problema de classificação de texto vem sendo estudado desde os anos 60, mas ganhou um interesse renovado com o avanço em hardwares, e principalmente pelo crescimento da Internet, que disponibiliza uma grande quantidade de informações com fácil acesso. Classificação de texto pode ser aplicada em muitos contextos, tais como: indexação e filtragem de documentos, geração automatizada de metadados e análise de sentimento (Manning e Schutze, 1999) (Sebastiani, 2002) (Pang e Lee, 2008).

O problema de classificação de texto pode ser resolvido com técnicas de aprendizagem de máquina, tal que um classificador é treinado com um conjunto de documentos previamente rotulados para aprender as características de cada classe de documentos. Essa abordagem busca substituir a classificação manual feita por especialistas, uma vez que não é necessária intervenção humana para construção do classificador (Sebastiani, 2002) (Forman, 2003).

Análise de Sentimento

Análise de sentimento, ou mineração de opinião, é uma área de pesquisa que analisa as opiniões, sentimentos, atitudes, avaliações e emoções de pessoas em relação a um determinado produto, serviço, organização, indivíduo, problema, evento ou tópico. É uma área de pesquisa que possui impacto em diversos domínios de aplicação. Opiniões são importantes para muitas atividades humanas porque são capazes de influenciar o comportamento. Na hora de tomar-se uma decisão, a opinião de outras pessoas é relevante (Pang e Lee, 2008) (Liu, 2012).

Muitas aplicações envolvem a análise de sentimento. Empresas têm interesse em opiniões de clientes sobre seus produtos e serviços, consumidores buscam opiniões antes de comprar um produto ou escolher um candidato na eleição. Antigamente, a opinião de indivíduos era obtida por pesquisas e entrevistas. Essa obtenção de opiniões públicas vem sendo usada em marketing, relações públicas e campanhas políticas há muito tempo, no entanto, essas pesquisas exigem muito tempo para ser finalizadas. Com o crescimento rápido de redes sociais, informações sobre diversos temas estão disponíveis em grande quantidade e com fácil acesso. Dessa maneira, sistemas automatizados de análise de sentimento são necessários (Liu, 2012).

Além de aplicações em diversos domínios, como serviços financeiros, eventos sociais, eleições políticas, marketing e serviços a clientes, análise de sentimento tem sido aplicada para diversos propósitos, como: predição de venda de produtos, relação entre apostas em esportes e opiniões em redes sociais, relação entre pesquisas eleitorais e sentimentos para previsão de resultado eleitoral, predição de desempenho de filmes e produtos e predição de mercado financeiro (Liu, 2012).

Análise de sentimento é uma subcategoria do problema de classificação de texto, com duas principais diferenças: problemas tradicionais de classificação de texto podem possuir diferentes números de classes a serem atribuídas, podendo variar de duas a milhares de classes. Por outro lado, em problemas de análise de sentimento, é muito comum que o número de classes varie de dois (positivo e negativo) para cinco (estrelas em uma avaliação) (Pang e Lee, 2008). Outra diferença importante é a relação entre as classes do problema: enquanto em problemas de classificação as palavras podem ser completamente não-relacionadas, em análise de sentimento o relacionamento das palavras é importante, pois uma palavra pode demonstrar sentimentos diferentes em diferentes contextos (Pang e Lee, 2008). Por esse motivo, a escolha da representação dos documentos é importante na análise de sentimento, tema que será abordado a seguir.

2.1.2 Representação Vetorial de Palavras

No domínio de NLP, e consequentemente em classificação de texto e análise de sentimento, é necessário converter documentos em vetores de características. A escolha de características tem impacto significativo na qualidade de um classificador, além de potencialmente diminuir o tamanho da base de treinamento sem comprometer o desempenho do classificador. Por isso, a representação de documentos de texto é um problema fundamental em NLP. Abordagens mais comuns costumam representar palavras vetorialmente (Forman, 2003) (Pang e Lee, 2008).

Os métodos mais simples e tradicionais de representação são os métodos *one-hot* e *bag-of-words*. No modelo *one-hot*, cada palavra é representada por um vetor de dimensão igual ao tamanho do vocabulário, tal que somente um índice do vetor possui valor um, responsável por representar a palavra, e todos os demais valores do vetor são iguais a zero, assim, um documento é representado pela concatenação dos vetores das palavras que compõe o documento. No modelo *bag-of-words*, cada documento é representado por um vetor de dimensão igual ao tamanho do vocabulário. Cada índice do vetor representa uma palavra, similar ao modelo *one-hot*, no

entanto, cada índice contém o número de ocorrências da palavra no documento, o que permite a representação do documento em somente um vetor. Ambas representações possuem problemas: a representação *one-hot* tem como problema central a dimensionalidade dos vetores, tal que para um vocabulário grande, cada documento seria representado pela concatenação de vários vetores de dimensão elevada. A representação *bag-of-words* tem como problema a falta de distinção entre os documentos, pois os vetores não contém nenhum tipo de informação referente à posição das palavras e suas relações (Forman, 2003) (Goldberg, 2016), o que torna essa representação inadequada para análise de sentimento. Essas representações são esparsas, por conterem informação relevante em poucas partes do vetor.

Uma alternativa a esse tipo de representação é a utilização de vetores densos, tal que cada índice do vetor corresponde a uma característica da palavra, contendo valores reais. Nesse tipo de representação, cada característica é incorporada em um espaço dimensional de tamanho pré-definido, e o conjunto de características é representado por um vetor nesse espaço. Esses vetores são chamados de *word embeddings*, e podem ser treinados com redes neurais artificiais, ou com métodos de decomposição de matrizes (Mikolov et al., 2013a) (Pennington et al., 2014) (Goldberg, 2016).

Em Mikolov et al. (2013b), foi demonstrado que *word embeddings* possuem grande capacidade de generalização, e são capazes de capturar padrões linguísticos e similaridades sintáticas e semânticas entre as palavras. Essas similaridades podem ser obtidas por expressões algébricas entre as palavras, como por exemplo: a soma dos vetores ”Alemanha” e ”capital” tem como vetor mais próximo a representação da palavra ”Berlim”. A Tabela 2.1 apresenta exemplos de relação entre palavras para diferentes domínios obtidas com o modelo de *word embeddings word2vec*, que será definido a seguir. Essas características podem impactar significativamente o desempenho de um classificador, pois o mesmo pode ser inicializado com parâmetros pré-definidos que representam as palavras.

Tabela 2.1: Exemplos de relação entre palavras de diferentes contextos obtidas com operações vetoriais utilizando o modelo *word2vec* (Mikolov et al., 2013a).

Relação	Exemplo 1	Exemplo 2	Exemplo 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Word2vec

Word2vec é um grupo de modelos utilizados para a obtenção de *word embeddings* por meio do uso de redes neurais. Os modelos do *word2vec* recebem um grande corpus de texto e produzem um espaço vetorial com muitas dimensões, tal que cada palavra do corpus é representada por um vetor no espaço. Os vetores são projetados no espaço vetorial, de forma que palavras que compartilhem semântica sejam posicionadas próximas no espaço (Mikolov et al.,

2013a). As duas arquiteturas que compõem o *word2vec*, *Continuous bag-of-words* (CBOW) e *Skip-gram* são apresentadas na Figura 2.1.

A arquitetura CBOW é treinada para prever a palavra baseada no contexto (palavras próximas) em que ela se encontra, enquanto a arquitetura *Skip-gram* é treinada para prever o contexto baseado na palavra de entrada. Ambas as arquiteturas do *word2vec* são redes neurais artificiais contendo uma única camada escondida, de modo que os pesos da camada escondida armazenam os valores das *word embeddings*, que são alterados durante o treinamento. O número de neurônios na camada escondida é igual a dimensão dos vetores de palavras, que é um parâmetro do algoritmo.

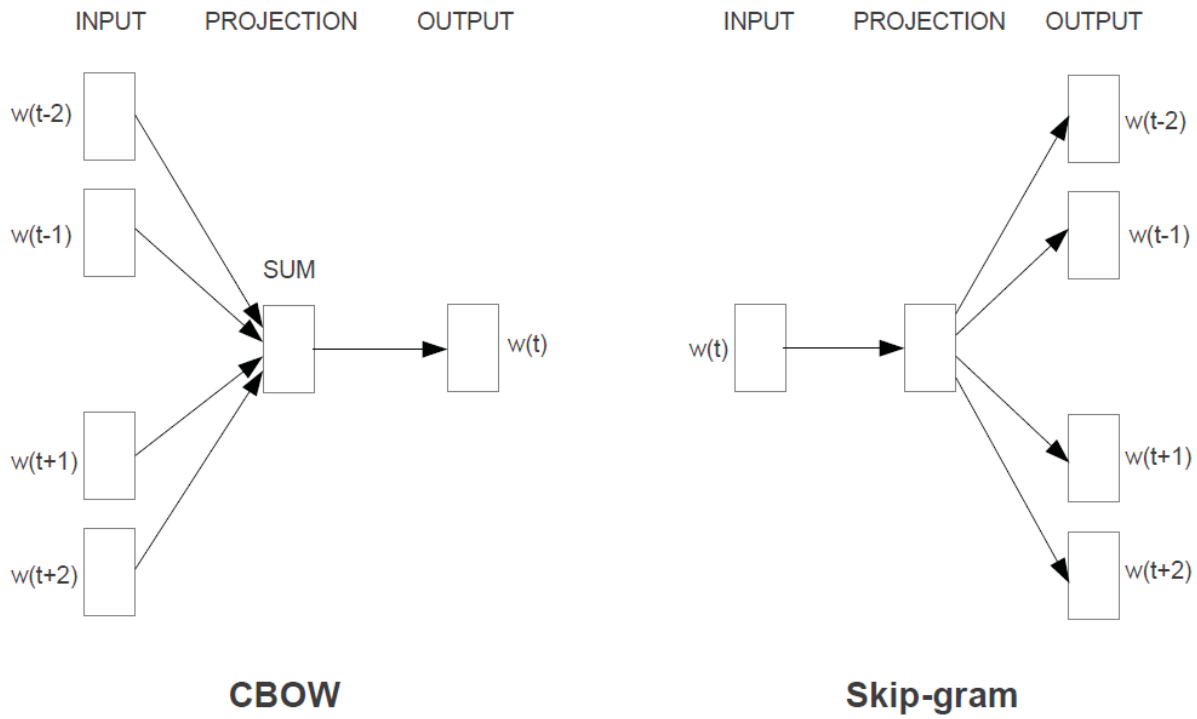


Figura 2.1: Arquiteturas CBOW e *Skip-gram* (Mikolov et al., 2013a).

Global Vectors

Global Vectors (*gloVe*) é um algoritmo não-supervisionado para obtenção de representações vetoriais de palavras (*word embeddings*), baseado em estatísticas extraídas de co-ocorrência de palavras. Enquanto o *word2vec* tem como objetivo de treinamento prever a palavra baseada em seu contexto (ou o contexto baseado na palavra), o *gloVe* treina seus vetores baseados em uma matriz de co-ocorrência de palavras, obtida por um corpus de treinamento que indica a frequência com que as palavras do corpus são correlacionadas entre si, ou seja, a frequência com que as palavras aparecem em um mesmo contexto (Pennington et al., 2014).

Dado um conjunto de documentos com vocabulário de tamanho V, a matriz de co-ocorrência de palavras de dimensões $V \times V$ é denotada por X, tal que X_{ij} representa o número de vezes que a palavra j aparece no mesmo contexto da palavra i. O contexto é definido como as N palavras que antecedem e sucedem a palavra i, onde N é uma janela de tamanho pré-definido. Seja $X_i = \sum_k X_{ik}$ o número de vezes que qualquer palavra aparece no contexto de i, e seja $P_{ij} = P(j|i) = X_{ij}/X_i$ a probabilidade de a palavra j aparecer no mesmo contexto que a palavra i (Pennington et al., 2014).

Considerando-se duas palavras, i e j, a relação entre elas em um determinado conceito pode ser medida por meio de várias palavras k, considerando a razão P_{ik}/P_{jk}: para palavras k que possuam relação com a palavra i, mas não com j, o valor da divisão será alto; para palavras k que possuam relação com j, mas não com i, o valor da divisão será baixo e; para palavras k que possuam relação, ou não, com i e j, o valor da divisão será próximo de um. A Tabela 2.2 apresenta exemplos de probabilidades e razões para diferentes palavras. Partindo do ponto de que vetores de palavras devem ser aprendidos a partir da razão entre as probabilidades de co-ocorrência entre as palavras, a forma genérica da função objetivo do *gloVe* é definida por:

$$F(w_i, w_j, \hat{w}_k) = \frac{P_{ik}}{P_{jk}}, \tag{2.1}$$

tal que w são vetores de palavras e $\hat{w}$ são vetores de palavras de contexto, de modo que F projete as palavras em um espaço vetorial com base na similaridade (P_{ik}/P_{jk}) entre elas. A Equação 2.1 é expandida em Pennington et al. (2014) para:

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^T \hat{w}_j + b_i + \hat{b}_j - \log X_{ij})^2, \tag{2.2}$$

tal que V é o tamanho do vocabulário, w_i, b_i e $\hat{w}_j$, $\hat{b}_j$ representam os vetores de palavra e *bias* das palavras i e j, respectivamente, X_{ij} representa o número de vezes que a palavra i aparece no contexto da palavra j, $f(x)$ representa uma função-peso, e J representa a função objetivo, que consiste em minimizar a diferença entre o produto escalar dos vetores das duas palavras e o logaritmo do número de suas co-ocorrências.

Tabela 2.2: Probabilidades para as palavras *ice* e *steam* com algumas palavras de contexto obtidas de um corpus com 6 bilhões de palavras (Pennington et al., 2014).

Probabilidade e Razão	k = *solid*	k = *gas*	k = *water*	k = *fashion*
$P(k\|ice)$	1.9 x 10^{-4}	6.6 x 10^{-5}	3 x 10^{-3}	1.7 x 10^{-5}
$P(k\|steam)$	2.2 x 10^{-5}	7.8 x 10^{-4}	2.2 x 10^{-3}	1.8 x 10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5 x 10^{-2}	1.36	0.96

2.2 *DEEP LEARNING*

2.2.1 Redes Neurais Artificiais

Redes Neurais Artificiais (RNAs) são classificadores projetados para simular o funcionamento do cérebro humano quando executa determinada tarefa. Uma RNA consiste de muitos nós simples, divididos em camadas, chamados neurônios: cada neurônio produz uma sequência de ativações. RNAs adquirem conhecimento por meio de um processo de aprendizagem, e esse conhecimento é armazenado em pesos sinápticos (pesos, ou parâmetros da rede), que conectam os diversos neurônios da rede. Neurônios da camada de entrada são ativados por meio da percepção dos dados de entrada, enquanto neurônios de camadas subsequentes são ativados pelo estímulo de camadas anteriores. O processo de aprendizagem é efetuado por meio de um algoritmo de aprendizagem, cuja função é modificar os pesos da rede com o objetivo de obter o comportamento desejado, por meio de treinamento (Haykin, 1999) (Schmidhuber, 2015).

Uma propriedade central de uma RNA é sua capacidade de aprender com base no ambiente, e melhorar seu desempenho por meio dessa aprendizagem. Esse aperfeiçoamento se

dá ao longo do tempo por meio do treinamento e, idealmente, a rede adquire mais conhecimento sobre o ambiente durante cada iteração do processo de aprendizagem (Haykin, 1999). Existem vários tipos de algoritmo de aprendizagem que podem ser aplicados à RNAs, sendo o mais utilizado o algoritmo de correção de erro (*error-correction*). Durante cada iteração do processo de treinamento, a rede é ativada com um exemplo de determinada classe do que deseja se classificar, e produz um sinal de saída. A saída produzida é aplicada em uma função objetivo que calcula o erro entre a saída e o resultado esperado. Em seguida, a rede neural ajusta seus parâmetros com o intuito de reduzir esse erro por meio de um algoritmo de correção de erro (LeCun et al., 2015).

Multilayer Perceptrons

Multilayer Perceptrons (MLPs) são redes neurais que consistem em um conjunto de nós conectados, constituídas em camadas, sendo elas: camada de entrada, uma ou mais camadas escondidas e camada de saída. Nesse tipo de rede, um sinal de entrada é propagado pela rede, camada à camada, tal que cada sinal de saída de um neurônio é aplicado em uma função de ativação não-linear. Em MLPs, o algoritmo de correção de erro mais comumente aplicado é o *backpropagation*. Esse algoritmo é composto essencialmente por duas fases:

(1) *Forward pass*: tal que um vetor de entrada estimula os nós da camada de entrada, e os sinais oriundos dessa camada são passados para as camadas subsequentes, até obter os sinais de saída. Cada nó z_j de uma camada l recebe estímulos de nós da camada antecedente e propaga os sinais recebidos para a camada subsequente depois de mapeá-los utilizando uma função de ativação. O sinal de saída y_j de um nó z_j é calculado segundo a Equação 2.3 (Hinton et al., 2012):

$$y_j^l = f(z_j^l), \qquad z_j^l = b_j^l + \sum_k w_{jk}^l \cdot y_k^{l-1}, \tag{2.3}$$

tal que l representa a camada atual, b_j^l representa o *bias* do j-ésimo neurônio da camada l, $l-1$ representa a camada anterior, w_{jk}^l é o peso da ligação entre o k-ésimo neurônio da camada $l-1$ e o j-ésimo neurônio da camada l, e $f(z_j)$ representa uma função de ativação. Durante essa etapa do algoritmo, os pesos da rede não são alterados. Uma ilustração do procedimento *forward pass* é apresentado na Figura 2.2. Após a obtenção dos sinais de saída, calcula-se o erro obtido pela rede baseado em uma função objetivo, necessário para aplicação da segunda fase do algoritmo.

(2) *Backward pass*: Nessa etapa, os pesos são ajustados de acordo com a saída produzida. Após a obtenção do erro, é feita uma propagação do erro nos pesos da rede, em sentido contrário a orientação da rede. Os pesos são ajustados para que a rede se adapte ao resultado esperado (Haykin, 1999). Uma ilustração do procedimento *backward pass* é apresentado na Figura 2.3.

O procedimento *backward pass* do algoritmo *backpropagation* computa o gradiente de uma função objetivo em relação aos parâmetros da rede por meio da aplicação da regra da cadeia para cálculo de derivadas. O sucesso do algoritmo se dá porque a derivada da função em relação à entrada de uma camada pode ser calculada usando a entrada da camada subsequente. O *backward pass* pode ser aplicado repetidamente para propagar gradientes por todas as camadas, começando pela camada de saída, até a camada de entrada. Após a computação dos gradientes, é feita a atualização dos pesos por meio de um algoritmo de atualização, como a descida de gradiente. O algoritmo aplica uma correção nos pesos da rede, subtraindo os valores dos parâmetros dos gradientes multiplicados por uma constante (parâmetro), chamada taxa de aprendizagem (*learning rate*) (Haykin, 1999) (LeCun et al., 2015) (Goodfellow et al., 2016).

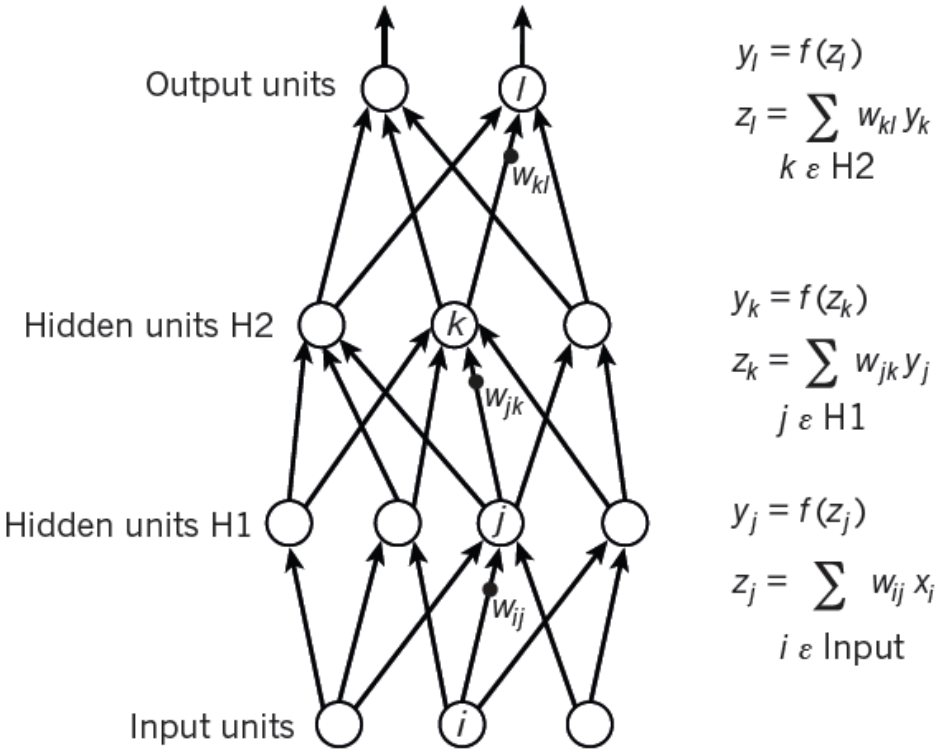


Figura 2.2: Exemplo de MLP com aplicação do algoritmo *forward pass*, contendo duas camadas escondidas, três neurônios de entrada e dois de saída. Em cada camada, é computada a entrada z, que é composta da soma da multiplicação dos pesos e das saídas da camada anterior. Em seguida, é aplicada uma função de ativação à z para obter a saída do neurônio (LeCun et al., 2015).

2.2.2 *Deep Neural Networks*

Deep Learning baseia-se na utilização de duas noções: experiência e a capacidade de enxergar o mundo como uma hierarquia de conceitos, tal que cada conceito é definido pela conexão de conceitos mais simples. Adquirir conhecimento por meio de experiência dispensa a necessidade de especificar formalmente o conhecimento necessário pelo computador. A noção de se aprender por experiência é aplicada em RNAs, que por sua vez serve como base para *deep learning*. A noção de hierarquia de conceitos permite que o computador aprenda conceitos complicados, construindo-os de forma mais simples, semelhante ao funcionamento do cérebro humano. *Deep learning* aplica a noção de hierarquia de conceitos introduzindo representações que são expressas por meio de representações mais simples: essa construção de conhecimento se dá por meio de várias camadas de representações, tal que cada camada do modelo descreve uma parte do conhecimento (Goodfellow et al., 2016). O modelo mais utilizado de *deep learning* são MLPs com muitas camadas, também chamadas de *deep neural networks* (DNNs). Uma DNN é uma MLP que possui mais de uma camada escondida, com muitos neurônios nessas camadas.

DNNs podem ser treinadas por meio do algoritmo *backpropagation*. DNNs com muitas camadas escondidas e muitos neurônios por camada são modelos muito flexíveis com um número muito grande de parâmetros. Isso permite a modelagem de relações muito complexas e não-lineares entre entrada e saída (Hinton et al., 2012).

DNNs exploram a propriedade de que muitos sinais naturais são hierarquias de composição, nas quais recursos de nível superior são construídos por meio da composição de níveis mais baixos. Em imagens, combinações de *pixels* formam bordas, bordas formam partes e partes

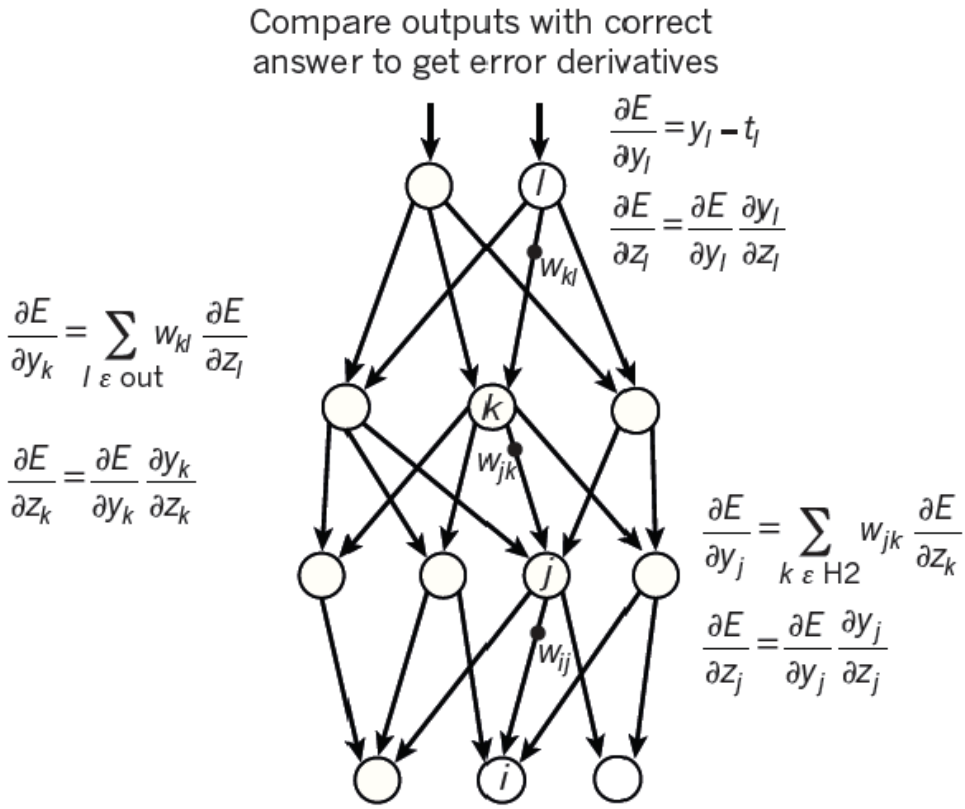


Figura 2.3: *Backward pass* do exemplo presente na Figura 2.2. (LeCun et al., 2015).

formam objetos. Em textos, essa composição também existe: textos são compostos por palavras, e processar textos em nível de palavras é muito similar ao processamento de imagens pixel a pixel (Cambria e White, 2014) (LeCun et al., 2015).

2.2.3 *Convolutional Neural Networks*

Convolutional Neural Networks (CNNs, Redes Neurais Convolucionais) são um tipo de arquitetura de DNNs que processam dados em forma de matrizes, sendo desenvolvidas para trabalhar com imagens (LeCun et al., 1989). CNNs vêm sendo aplicadas com sucesso em diversas áreas, como imagens, áudio e também textos. Nos últimos anos, surgiu um renovado interesse nas CNNs devido à grande capacidade de processamento das GPUs atuais, pela disponibilidade de bases de treinamento maiores e pela obtenção de resultados melhores que outros métodos (Krizhevsky et al., 2012) (Zeiler e Fergus, 2013).

O nome "Redes Neurais Convolucionais"se dá pela aplicação de operações chamadas de convoluções. CNNs são DNNs que aplicam convolução no lugar de multiplicação de matrizes em pelo menos uma das camadas (Goodfellow et al., 2016). A operação de convolução consiste em somar as multiplicações elemento a elemento de um filtro (*kernel*) por uma matriz, conforme definido na Equação 2.4:

$$(I \otimes K)_{ij} = \sum_{m=0}^{k_1-1}\sum_{n=0}^{k_2-1} I(i+m, j+n) \cdot K(m,n) \tag{2.4}$$

tal que $\otimes$ representa a operação de convolução, I representa a matriz de entrada, K representa o filtro de convolução (*kernel*), i e j representam as posições da matriz I, e k_1 e k_2 representam

as dimensões do *kernel*. A saída de uma convolução é usualmente chamada de *feature map*. A Figura 2.4 apresenta uma ilustração de uma operação de convolução, conforme usada em CNNs.

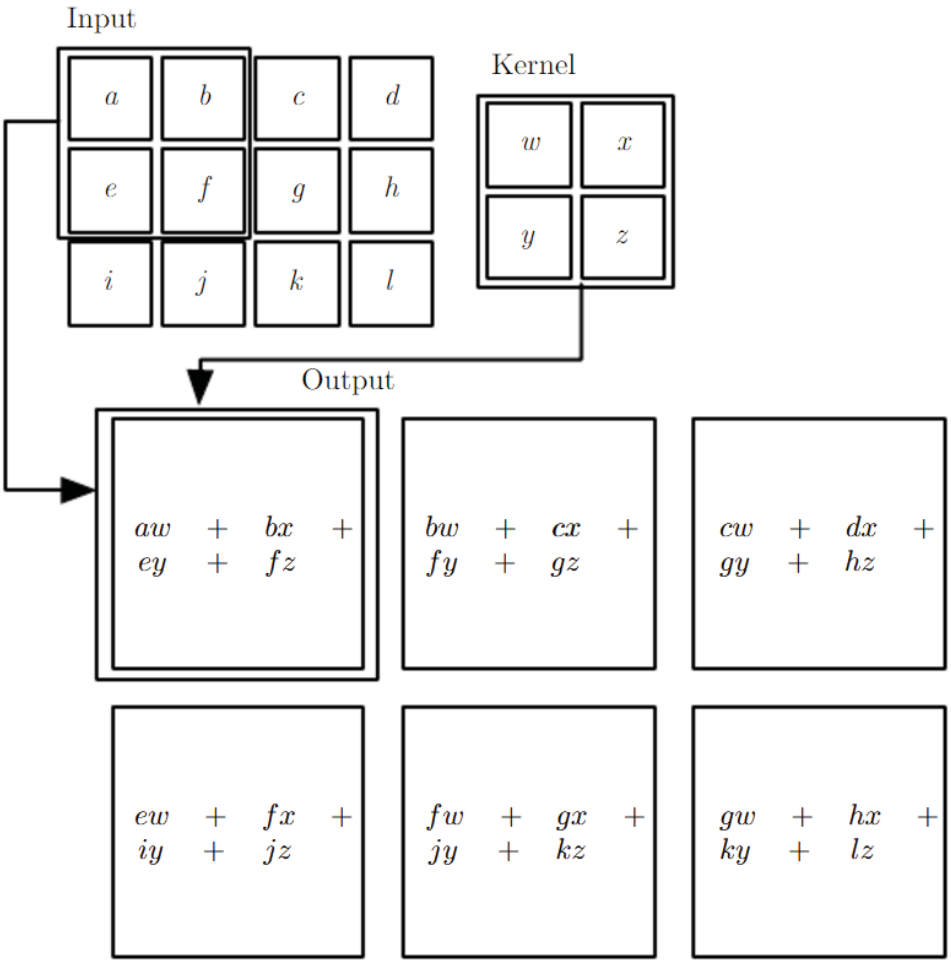


Figura 2.4: Exemplo de convolução 2D (Goodfellow et al., 2016).

Convolução incorpora três características importantes que podem contribuir significativamente para o desempenho de um sistema: pesos esparsos, compartilhamento de parâmetros e representações invariantes (LeCun et al., 2015) (Goodfellow et al., 2016).

- Pesos esparsos: Com o uso de filtros menores que o tamanho da entrada, é possível detectar características importantes, como bordas em imagens enormes, armazenando-se menos parâmetros. Isso permite a redução do uso de memória e a realização de menos operações na saída.

- Compartilhamento de parâmetros: Como consequência de pesos esparsos, CNNs apresentam compartilhamento de parâmetros. Em uma rede neural tradicional, cada elemento da matriz de pesos é usado exatamente uma vez ao calcular a saída de uma camada, sendo multiplicado por um elemento da entrada e não sendo reutilizado. Em uma CNN, cada elemento do filtro é usado em todas as posições da entrada. O compartilhamento de parâmetros usado pela operação de convolução significa que, em vez de aprender conjuntos separados de parâmetros, aprende-se somente um conjunto.

- Representações invariantes: Em CNNs, compartilhamento de parâmetros faz com que a camada apresente invariância à posição. Em matrizes como imagens, grupos de

elementos próximos são muitas vezes altamente correlacionados, formando conjuntos locais distintos que são facilmente detectados. As estatísticas locais de imagens e outros sinais são invariantes à posição: se um conjunto pode aparecer em uma parte da imagem, pode aparecer em qualquer lugar, o que torna atrativa a ideia de conjuntos em diferentes locais compartilhando os mesmos pesos e detectando o mesmo padrão em diferentes partes da matriz.

Em CNNs, uma camada usualmente contém três etapas, conforme apresentado na Figura 2.5. Na primeira etapa, são executadas diversas convoluções em paralelo, com o intuito de produzir um conjunto de ativações lineares. Na segunda etapa, é executada uma função de ativação não-linear sobre cada ativação linear. Na terceira etapa, é aplicada uma função de agrupamento (*pooling*) para modificar a saída da camada.

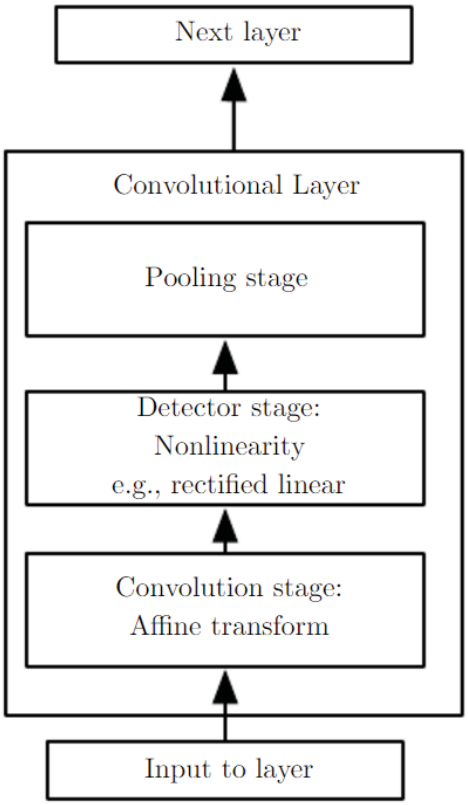


Figura 2.5: Etapas de uma camada de uma CNN (Goodfellow et al., 2016).

O propósito da camada de *pooling* é juntar recursos semanticamente semelhantes na saída de uma determinada camada da rede. A camada de *pooling* essencialmente reduz a imagem por meio da união de valores vizinhos, de modo que as camadas subsequentes trabalhem com representações de menor dimensão, invariantes a pequenas mudanças e distorções. A operação mais usual é a *max pooling*, que retorna o maior valor de um conjunto de valores que se encontram em uma janela (LeCun et al., 1998) (Johnson e Zhang, 2015a) (LeCun et al., 2015) (Goodfellow et al., 2016). A Figura 2.6 apresenta um exemplo de *max pooling* com janela de tamanho dois.

Usualmente na última camada de uma CNN, é aplicada uma função *softmax*, apresentada na Equação 2.5. Essa função é utilizada para prever a probabilidade associada a uma distribuição,

recebendo um vetor com valores reais, transformando-o em um vetor de valores entre 0 e 1, tal que a soma de seus elementos é igual a 1 (Goodfellow et al., 2016).

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)} \tag{2.5}$$

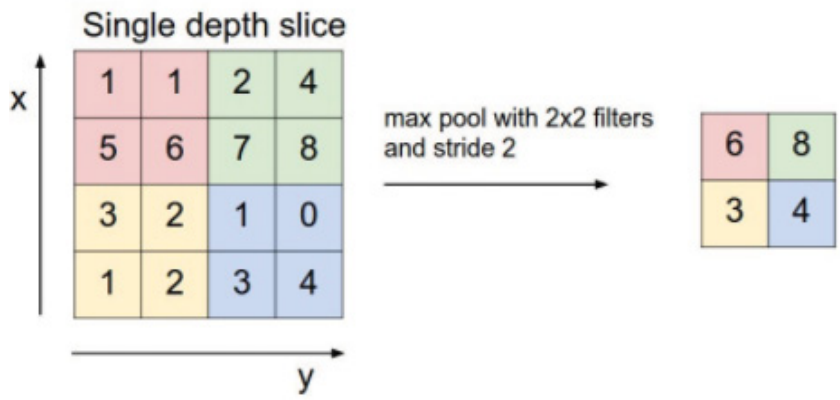


Figura 2.6: Operação de *max pooling* com janela 2x2.
Fonte: `http://cs231n.github.io/convolutional-networks/`

2.3 ALGORITMOS DE AGRUPAMENTO

Algoritmos de agrupamento (*clustering*) são técnicas utilizadas para resolver problemas em ambientes não-supervisionados. Em problemas não-supervisionados, os dados disponíveis não são rotulados, ou seja, a classe dos dados não é conhecida. O objetivo de um algoritmo de agrupamento é particionar um conjunto de objetos em grupos (*clusters*) baseado em uma medida de similaridade pré-definida, de forma que objetos que possuam características semelhantes sejam colocados no mesmo grupo, e objetos que possuam características diferentes sejam atribuídos a grupos diferentes (Jain e Dubes, 1988) (Manning e Schutze, 1999) (Duda et al., 2001) (Xu e Wunsch, 2005).

Clustering vem sendo utilizado em uma ampla variedade de campos, como aprendizagem de máquina e reconhecimento de padrões, inteligência artificial, análise de banco de dados, coleta de documentos textuais, segmentação de imagens, além de outras áreas de conhecimento, como medicina, biologia, geografia, geologia, ciências sociais, engenharias e economia (Xu e Wunsch, 2005).

Algoritmos de agrupamento são usualmente divididos em duas categorias: hierárquico e não-hierárquico. Um agrupamento hierárquico é uma hierarquia, tal que cada nó representa um sub-*cluster* do nó ascendente. Cada nó representa o *cluster* que contém todos os objetos de seus descendentes. Agrupamentos não-hierárquicos consistem na divisão de objetos em um número determinado de *clusters*, e usualmente, cada objeto pertence somente a um *cluster*. Algoritmos de agrupamento não-hierárquico são em sua maioria iterativos, começando com um conjunto inicial de *clusters* e melhorando iterativamente por meio da relocação que redistribui os objetos entre os *clusters*. Um exemplo de agrupamento não-hierárquico é o algoritmo *k-means*, que é apresentado no Algoritmo 1 (Manning e Schutze, 1999) (Xu e Wunsch, 2005).

Um aspecto fundamental em algoritmos de agrupamento é a escolha da medida de similaridade entre os objetos. Um objeto em um conjunto de dados é descrito por uma sequência de características, que normalmente é representada por um vetor. Para medir a proximidade entre

vetores, podem ser usadas funções de distância, que capturam características contínuas. Nesse contexto, a métrica mais utilizada é a distância Euclidiana, sendo aplicada em vários algoritmos, incluindo o algoritmo *k-means* (Xu e Wunsch, 2005). O cálculo da distância Euclidiana é apresentado na Equação 2.6.

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ... + (x_n - y_n)^2} \tag{2.6}$$

tal que n representa a dimensão dos vetores x e y e $d(x, y)$ representa a distância Euclidiana entre os vetores.

Algoritmo 1 Algoritmo *k-means*:

```
Dado um conjunto X = {x_1, ...x_n}
Escolher um valor para k
Escolher aleatoriamente k pontos de X como centroides
while critério de parada do
    for all x_i ∈ X do
        Calcular distância entre x_i e centroides
        Atribuir x_i ao centroide mais próximo
    end for
    Recalcular centroide para cada cluster
end while
```

2.3.1 Agrupamento Baseado em Densidade

Embora algoritmos de agrupamento hierárquicos e não-hierárquicos sejam efetivos em muitos casos, para a resolução de problemas com bases de dados grandes o algoritmo deve satisfazer os seguintes requisitos: (1) Conhecimento mínimo do domínio da aplicação para determinação de parâmetros de entrada, uma vez que valores apropriados não são conhecidos inicialmente em grandes bases de dados; (2) Capacidade de descoberta de *clusters* de formato arbitrário; (3) Boa eficiência computacional. Nesse contexto, algoritmos de agrupamento baseados em densidade podem ser utilizados (Ester et al., 1996) (Hinneburg et al., 1998).

Em algoritmos de agrupamento baseados em densidade, elementos vizinhos são agrupados em *clusters* com base em condições locais, como a densidade local dos pontos, o que permite a detecção de *clusters* de formato arbitrário. Essas informações locais são obtidas em uma única leitura da base de dados, proporcionando boa eficiência computacional e consequentemente, permitindo sua aplicação em grandes conjuntos de dados, exigindo poucos parâmetros para aplicação do algoritmo (Ester et al., 1996) (Hinneburg et al., 1998) (Rodriguez e Laio, 2014).

Em Ester et al. (1996), foi proposto o algoritmo DBSCAN (*Density-based spatial clustering of applications with noise*), que é muito utilizado em grandes conjuntos de dados (Xu e Wunsch, 2005). Nesse algoritmo, define-se um limiar de densidade e para cada ponto de um *cluster*, a densidade de cada ponto na vizinhança deve exceder esse limiar para que o ponto seja incluído no *cluster*. Pontos com densidades inferiores a esse limiar são descartados.

Mais recentemente, em Rodriguez e Laio (2014), foi proposto um algoritmo de agrupamento baseado em densidade com comportamento similar ao algoritmo *k-means*, por basear-se somente na distância entre os pontos. Esse algoritmo parte da premissa de que centroides são cercados por vizinhos que possuem densidade local menor, e que estão a uma distância relativamente grande de qualquer ponto com densidade maior. Para cada ponto i, são

computados dois valores: a densidade local ρ_i e a distância δ_i de pontos com maior densidade. A densidade local ρ_i é definida conforme a Equação 2.7:

$$\rho_i = \sum_j \chi(d_{ij} - d_c) \tag{2.7}$$

tal que d_{ij} representa a distância entre os pontos i e j, d_c é uma distância de corte, e

$$\chi(j) = \begin{cases} 1 & \text{se } d_{ij} < d_c \\ 0 & \text{se } d_{ij} \geq d_c \end{cases} \tag{2.8}$$

ou seja, ρ_i é igual ao número de pontos que são mais próximos de d_c ao ponto i.

A distância δ_i é medida pelo cálculo da distância mínima entre o ponto i e qualquer outro ponto com maior densidade, conforme a Equação 2.9:

$$\delta(i) = min_{j:\rho_j > \rho_i}(d_{ij}) \tag{2.9}$$

Dessa forma, centroides são os pontos que possuem altos valores de δ_i.

2.3.2 Detecção de *Outliers*

Um *outlier* em um conjunto de dados pode ser definido informalmente como uma observação que é consideravelmente diferente dos demais dados. *Outliers* surgem devido a falhas mecânicas, alterações no comportamento do sistema, comportamento fraudulento, erro humano, ou simplesmente por meio de desvios naturais nas populações. A detecção de *outliers* tem sido utilizada há séculos para detectar e, quando apropriado, remover essas observações anômalas do conjunto de dados (He et al., 2003) (Hodge e Austin, 2004).

A detecção de *outliers* é uma importante área de pesquisa no mundo da mineração de dados com inúmeras aplicações, como detecção de fraudes, detecção de invasão, previsão do tempo, marketing, monitoramento de atividades, monitoramento de condições médicas, detecção de novidades em imagens e textos, detecção de entidades inesperadas em bancos de dados e detecção de dados incorretos em bases de treinamento. A detecção e remoção de *outliers* permite a purificação da base de dados para o processamento (He et al., 2003) (Hodge e Austin, 2004).

Para resolução do problema de detecção de *outliers*, três abordagens costumam ser utilizadas: detecção de novidade, classificação supervisionada e agrupamento não-supervisionado. Técnicas de detecção de novidade são abordagens semi-supervisionadas, pois necessitam de dados pré-classificados, classificando os dados em somente uma classe, que fornece o modelo de normalidade. Essa técnica é capaz de aprender o modelo de forma incremental à medida que novos dados chegam, adaptando o modelo conforme a disponibilidade de um novo exemplo. O objetivo de detecção de novidade é definir um limite de normalidade. Esse tipo de abordagem reconhece um exemplo como normal se ele se encontra dentro da classe e reconhece dados como *outliers* caso contrário (Hodge e Austin, 2004).

Detecção de novidade com classificação supervisionada consiste na aplicação de um classificador, como uma rede neural, treinado com dados rotulados divididos em classes: normais e *outliers*. A região que se encontra fora da classe normal representa a classe de *outliers* (Hodge e Austin, 2004).

Em agrupamento não-supervisionado, os *outliers* são determinados sem conhecimento prévio dos dados. Esse tipo de abordagem identifica os pontos mais remotos e os indica como potenciais *outliers*, assumindo que erros ou falhas são separados dos dados normais. O uso de algoritmos de agrupamento permite simultaneamente a tarefa de classificação e detecção

de *outliers*: a região considerada normal pode ser subdividida em categorias, o que permite a classificação, e elementos fora da região são considerados *outliers* (Hodge e Austin, 2004).

2.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou os principais conceitos relacionados ao presente trabalho: Processamento de Linguagem Natural, destacando o problema de análise de sentimento e representação vetorial de palavras, *Deep Learning* e Redes Neurais Convolucionais, e algoritmos de agrupamento, incluindo algoritmos de agrupamento baseados em densidade e detecção de *outliers*.

No contexto deste trabalho, as representações vetoriais descritas, *word embeddings*, são utilizadas como representação dos documentos para aplicação no classificador utilizado, uma CNN, e no algoritmo de agrupamento baseado em densidade, com o intuito de extrair informações do espaço semântico. No próximo capítulo são apresentados os trabalhos relacionados ao presente trabalho.

3 TRABALHOS RELACIONADOS

Nesse capítulo são apresentados os trabalhos relacionados que possuem conceitos e técnicas semelhantes ao presente trabalho. Embora existam muitos trabalhos que aplicam CNNs a diversos problemas da área de NLP, serão apresentados trabalhos que aplicam CNNs aos problemas de classificação de texto e análise de sentimento. No final do capítulo, será apresentada uma análise crítica dos principais trabalhos relacionados.

Em Kalchbrenner et al. (2014), foi proposta uma CNN para modelagem semântica de frases, chamada *Dynamic Convolutional Neural Network* (DCNN). O nome se dá pela aplicação de uma operação de *pooling* dinâmica (*dynamic k-max pooling*), que consiste em selecionar os k maiores valores de uma sequência, variando o valor de k de acordo com a camada da rede e o tamanho da frase.

A arquitetura possui quatro camadas, com duas camadas convolucionais: na primeira camada, as frases são representadas por meio de *word embeddings* (Turian et al., 2010); na segunda e terceira camadas, as informações de camadas anteriores são aplicadas em diversos filtros, seguidos da operação de *pooling* dinâmico e funções não-lineares e; na quarta camada, as informações da terceira camada são completamente conectadas a uma *softmax* para prever a distribuição de probabilidade para cada classe do problema. A DCNN foi testada em quatro bases de dados, sendo três delas de análise de sentimento e obteve resultados, em média, 2,8% superiores para análise de sentimento em comparação com trabalhos anteriores (Go et al., 2009) (Socher et al., 2013).

Em Dos Santos e Gatti (2014), foi proposta uma CNN que utiliza representações de frases por meio de palavras e também por seus caracteres, chamada *Character to Sentence Convolutional Neural Network* (CharSCNN). Nessa CNN, os vetores de palavra são compostos pelas *word embeddings* e *character-level embeddings* da palavra. A primeira camada da rede transforma as palavras em *word embeddings*, com o algoritmo *skip-gram* do *word2vec*. Em seguida, cada caractere da palavra é representado por um *character-level embedding* de tamanho fixo, e a informação de caractere da palavra é representado pela combinação de todos os *character-level embeddings*.

A motivação da representação em nível de palavras é a captura de informações sintática e semânticas, enquanto a motivação da representação em nível de caracteres é a captura de informações morfológicas. Essas informações são passadas para duas camadas de convolução. A CharSCNN foi testada em duas bases de dados de análise de sentimento, obtendo resultados, em média, 1,5% superiores em comparação com outros algoritmos (Go et al., 2009) (Socher et al., 2013).

Em Kim (2014), foi proposta uma CNN com três camadas, com uma camada de convolução. A característica principal da arquitetura é a existência de dois canais de entrada, o que permite a aplicação simultânea de *word embeddings* estáticas e treináveis, ou seja, em um dos canais da rede as *word embeddings* são treinadas como parâmetros da rede, o que permite sua atualização para uma tarefa específica e no outro canal elas são mantidas conforme sua inicialização.

Com o objetivo de testar a capacidade da arquitetura, os autores conduziram experimentos com quatro modelos de representação sobre sete bases de dados. No primeiro modelo, as palavras são inicializadas aleatoriamente e atualizadas durante o treinamento. No segundo modelo, as palavras são inicializadas com *word embeddings* pré-treinadas com o modelo CBOW do *word2vec* e mantidas estáticas durante o treinamento (modelo estático). No terceiro modelo, inicializado de maneira similar ao segundo, as *word embeddings* são atualizadas durante o treinamento (modelo treinável). No quarto modelo, é feita uma combinação dos modelos dois e três, utilizando dois canais de entrada.

Os resultados dos experimentos sugerem que vetores pré-treinados são fundamentais para aumento de desempenho e possuem impacto em diversos domínios. Além disso, o trabalho mostrou que o uso de *word embeddings* treináveis pode levar a um aumento na taxa de acerto da rede, uma vez que essas representações são adaptadas para uma tarefa específica.

Em Johnson e Zhang (2015a), foi proposta uma CNN com uma camada de convolução, utilizando dois modelos de representação de documentos diferentes com o intuito de investigar a efetividade de CNNs sem a necessidade de recursos adicionais (uso de *word embeddings*). No primeiro modelo, os dados são representados por meio da representação *one-hot*. No segundo modelo, é utilizada a representação *bag-of-words*, na tentativa de diminuir a dimensionalidade dos vetores da representação *one-hot*.

As duas variações foram testadas em três bases de dados, sendo duas delas de análise de sentimento. Os resultados obtidos demonstram que a arquitetura teve resultados comparáveis com resultados de arquiteturas com aplicação de *word embeddings* (Kalchbrenner et al., 2014) (Kim, 2014). Para as bases de dados de análise de sentimento, os melhores resultados foram obtidos com a representação *one-hot*, enquanto para a base de dados de categorização de texto os melhores resultados foram obtidos com a representação *bag-of-words*.

Em Johnson e Zhang (2015b), os autores complementam o trabalho feito em Johnson e Zhang (2015a) adicionando um segundo canal de entrada para complementar as informações obtidas pela representação *one-hot*, utilizando *region embeddings*. Diferente de *word embeddings*, *region embeddings* são treinadas com o objetivo de prever o contexto de uma região de palavras, tais que a informação armazenada nos vetores passados para a CNN se referem a mais de uma palavra. A representação de uma região de palavras é motivada pela aprendizagem de conceitos importantes para uma tarefa específica.

Os autores executaram experimentos nas três bases de dados utilizadas em Johnson e Zhang (2015a). Foram executados testes com três variações de *region embeddings*, com diferentes tamanhos de *n*-gramas[1] e dimensões de vetores. Os autores reportaram melhores resultados para todas as bases de dados em comparação com Kim (2014) e Johnson e Zhang (2015a), e observaram que os melhores resultados foram obtidos concatenando as três *region embeddings*, com tamanho 100 cada.

Em Wang et al. (2015), foi proposta uma CNN para classificação de texto, contendo quatro camadas, sendo uma delas de convolução. A arquitetura conta com a aplicação de um algoritmo de agrupamento baseado em densidade (Rodriguez e Laio, 2014) para encontrar cliques semânticos no espaço vetorial.

Na primeira camada da rede, as palavras são inicializadas por meio de *word embeddings*, formando as matrizes de frases. Na segunda camada, o algoritmo de agrupamento baseado em densidade é utilizado para a formação de cliques semânticos. Após a obtenção dos cliques semânticos, as palavras da frase são combinadas em *n*-gramas, sendo *n* um parâmetro do algoritmo, por meio da soma dos vetores das palavras, formando unidades semânticas. Após a formação das unidades semânticas, calcula-se a distância Euclidiana de cada unidade semântica

[1]Sequência de *n* elementos consecutivos no documento.

para cada clique semântico. Caso a distância seja menor que um limiar pré-definido, a unidade semântica é selecionada para constituir a matriz semântica da frase. Na terceira camada, as matrizes de frases e semânticas são combinadas e aplicadas em filtros de convolução, seguidos da operação *k-max pooling*. Na quarta camada, os dados são aplicados em uma função *softmax*, responsável pela classificação.

Os autores conduziram experimentos em duas bases de dados, ambas de classificação de texto. A inicialização das palavras foi testada com três *word embeddings* diferentes (Collobert et al., 2011) (Mikolov et al., 2013a) (Pennington et al., 2014), e foram testados diferentes tamanhos de janela para cálculo das unidades semânticas. Os resultados obtidos com as três inicializações foram muito próximos, e o tamanho da janela teve impacto significativo no desempenho: tamanhos de janela pequenos podem causar perda de informações importantes, enquanto tamanhos de janela grandes podem introduzir ruído à CNN.

Em Severyn e Moschitti (2015), foi proposta uma CNN com três camadas, com uma camada de convolução, para análise de sentimento do *Twitter* com arquitetura similar ao trabalho de Kim (2014). A diferença principal do trabalho é a manipulação no treinamento das palavras para uso na CNN, com o intuito de inicializar a rede com bons parâmetros.

Na tentativa de melhorar os parâmetros iniciais da rede, os autores propuseram três etapas para inicialização. Na primeira etapa, as palavras foram inicializadas com *word embeddings* usando o algoritmo *skip-gram* do *word2vec*, treinados com cinquenta milhões de *tweets*. Na segunda etapa, para capturar comportamento de sentimento as *word embeddings* foram refinadas utilizando dez milhões de *tweets* rotulados em positivo e negativo. Na terceira etapa, os parâmetros obtidos na etapa anterior foram passados como entrada para a primeira camada da rede.

Os autores executaram experimentos em cinco bases de dados diferentes com três inicializações de parâmetros: aleatoriamente e não-supervisionada, similar à inicialização feita em Kim (2014) e efetuando as três etapas propostas. Os resultados demonstraram que o uso de um algoritmo não-supervisionado para inicialização das palavras (*skip-gram*) melhorou significativamente os resultados obtidos, e o refinamento das representações supervisionadamente para a tarefa específica de análise de sentimento trouxe maiores ganhos de desempenho em termos de taxa de acerto.

Em Zhang et al. (2015), foi proposta uma CNN que extrai informações diretamente de caracteres, sem o uso de *word embeddings* ou representações de palavras como *one-hot* e *bag-of-words*. A arquitetura possui nove camadas, sendo seis camadas de convolução e três camadas completamente conectadas.

Na camada de entrada, o documento é convertido em uma sequência de vetores de caractere. Os vetores são representados por *one-hot*, tais que cada vetor possui dimensão de tamanho 70, que representa o número de caracteres do alfabeto. A camada de entrada recebe a concatenação dos vetores de caracteres para os últimos 1014 caracteres do texto. Caracteres inexistentes no alfabeto são representados por vetores com todos os elementos iguais a zero. As seis camadas de convolução são compostas por convoluções unidimensionais, seguidas por uma função não-linear e o operador *max-pooling*. As três camadas subsequentes são completamente conectadas, seguidas por uma *softmax*, responsável pela classificação.

Para aumentar o número de dados disponíveis, os autores utilizaram um dicionário para substituir palavras por seus sinônimos nos documentos. As palavras que possuem sinônimos foram extraídas dos documentos, e foram selecionadas aleatoriamente r palavras para serem substituídas por seus sinônimos.

Para avaliar o modelo proposto, os autores construíram oito bases de dados, sendo quatro de análise de sentimento. Para comparação, os autores implementaram diferentes arquiteturas

com diferentes representações, como *bag-of-words*, TFIDF e *word2vec*. A arquitetura proposta encontrou os melhores resultados para quatro bases de dados.

Em Conneau et al. (2016), os autores propuseram uma arquitetura com muitas camadas escondidas, contendo até 29 camadas convolucionais. Na camada de entrada, os documentos são inicializados pela concatenação de vetores de caractere, tais que cada caractere do texto é representado por um vetor de tamanho fixo.

A primeira camada convolucional aplica 64 convoluções de tamanho 3. Em seguida, são aplicados diversos blocos convolucionais, tais que cada bloco aplica diferentes quantidades de convoluções, todas com tamanho 3, seguidas de uma função não-linear. Cada bloco convolucional possui duas camadas convolucionais. Os blocos convolucionais são seguidos de operações de *pooling*. O número de blocos convolucionais varia de acordo com a arquitetura. Os dados são passados para uma camada de *k-max pooling*, seguida de três camadas completamente conectadas e uma *softmax*.

Os autores executaram testes com 4 quantidades de camadas convolucionais: 9, 17, 29 e 49. Os experimentos foram executados nas bases de dados apresentadas em Zhang et al. (2015), sem o uso de *data augmentation*[2] e pré-processamento. Os resultados obtidos foram melhores para as oito bases de dados em comparação com a arquitetura de Zhang et al. (2015), e sugerem que a profundidade da CNN teve impacto no desempenho em termos de taxa de acerto, visto que os melhores resultados foram obtidos com 29 camadas. No entanto, o aumento na quantidade de camadas degradou o desempenho da rede: a taxa de acerto com 49 camadas convolucionais foi pior do que com 29 camadas.

3.1 DISCUSSÃO

O presente trabalho tem maior proximidade com os trabalhos de Kalchbrenner et al. (2014), Kim (2014), Wang et al. (2015) e Severyn e Moschitti (2015). Os trabalhos de Kalchbrenner et al. (2014) e Kim (2014) propõem CNNs com poucas camadas mas que são capazes de obter bons resultados e inspiraram muitos trabalhos na área de NLP. Além disso, o trabalho de Kim (2014) demonstra que uma boa representação de palavras é muito importante para obtenção de melhores resultados.

O trabalho de Wang et al. (2015) utiliza o algoritmo baseado em agrupamento baseado em densidade de Rodriguez e Laio (2014) para encontrar cliques semânticos e introduzir informações complementares na CNN. O principal problema com essa abordagem é a possível introdução de ruídos pela combinação de palavras.

O trabalho de Severyn e Moschitti (2015), por sua vez, também busca introduzir mais conhecimento para a CNN antes do treinamento por meio do refinamento da representação utilizando-se uma base de dados de análise de sentimento. O problema do refinamento das palavras é o tempo de execução devido ao tamanho da base utilizada, além de ser necessário o refinamento para aplicações específicas.

Os trabalhos de Johnson e Zhang (2015a) e Johnson e Zhang (2015b) são importantes por demonstrarem a capacidade das CNNs de obter bons resultados independentemente da representação utilizada. No entanto, o uso de representações *one-hot* e *bag-of-words* não são capazes de capturar semântica do texto, além de possuírem maior dimensionalidade em comparação com outras representações.

Os trabalhos de Dos Santos e Gatti (2014), Zhang et al. (2015) e Conneau et al. (2016) utilizam representações por meio de caracteres. Esses trabalhos obtiveram ótimos resultados com diferentes números de camadas e demonstram que esse tipo de representação captura informações

[2]Transformações nos dados para aumentar a base de dados.

morfológicas das palavras. No entanto, o uso desse tipo de representação requer uma grande quantidade de exemplos para treinamento da rede de forma que a mesma represente a sequência de caracteres satisfatoriamente.

3.2 CONSIDERAÇÕES FINAIS

Neste capítulo foram apresentados os trabalhos relacionados ao presente trabalho. Esses trabalhos utilizam CNNs para resolução dos problemas de categorização de texto e análise de sentimento, e os resultados sugerem que as representações utilizadas têm impacto significativo na taxa de acerto do classificador. Esses resultados motivaram o desenvolvimento deste trabalho, que consiste na detecção de *outliers* no espaço semântico por meio da aplicação de um algoritmo de agrupamento baseado em densidade sobre representações vetoriais de palavras com o intuito de melhorar as representações de documentos. A abordagem proposta é apresentada no próximo capítulo.

4 ABORDAGEM PROPOSTA

Para a resolução do problema de análise de sentimento, a escolha da representação das palavras é crucial para o desempenho do classificador, em termos de taxa de acerto e eficiência computacional. A representação vetorial por meio de *word embeddings* é uma ótima escolha, dado que os vetores que representam as palavras são treinados considerando o contexto das palavras no documento, em que os vetores possuem tamanho pré-definido e cada índice do vetor contém uma característica da palavra, além de ser capaz de manter a ordem das palavras, característica importante para problemas de análise de sentimento.

As representações vetoriais por *word embeddings* possuem uma grande capacidade de generalização e são capazes de capturar padrões linguísticos e similaridades sintáticas e semânticas entre as palavras. Conforme mencionado no Capítulo 2, em Mikolov et al. (2013a) e Mikolov et al. (2013b) foi demonstrado que os modelos CBOW e *skip-gram* do *word2vec* capturam relações entre as palavras que podem ser obtidas por meio de operações aritméticas vetoriais simples, como soma de vetores. Essas relações se dão por meio do treinamento, que considera o contexto das palavras, como no modelo *skip-gram* que tenta prever as palavras em volta da palavra na frase. Assim, palavras que aparecem juntas frequentemente em várias frases possuem relação no espaço vetorial.

As características obtidas pelas *word embeddings* sugerem que as palavras que possuem significado parecido e compartilham semântica são projetadas próximas no espaço vetorial, uma vez que aparecem repetidamente no mesmo contexto. Dado que a representação das palavras possui grande importância na qualidade de um classificador, e que a representação por *word embeddings* projeta as palavras em um espaço vetorial semântico, a aplicação de algoritmos de agrupamento sobre os vetores que representam as palavras na tentativa de contribuir na representação serviu como premissa para a proposta deste trabalho.

Problemas de alta dimensionalidade são muito desafiadores, o que torna muito importante a escolha de um bom algoritmo de agrupamento. Nesse contexto, algoritmos de agrupamento baseados em densidade são uma ótima opção, por conta de seu desempenho computacional e sua capacidade de detectar *clusters* com formatos arbitrários.

Usualmente, o significado dos documentos é dado por algumas palavras-chave ou frases-chave. Em problemas de análise de sentimento, é possível que o documento possua palavras irrelevantes para a classificação, e essas palavras apareçam em diversos documentos. Sendo assim, podem ser consideradas como *outliers* as palavras que se encontram em regiões de baixa densidade, ou seja, regiões que contém palavras que não são similares a outras palavras no espaço semântico, e algoritmos de detecção de *outliers* podem ser utilizados para remover essas palavras dos documentos sem perder seu significado.

Para detecção de *outliers*, conforme apresentado no Capítulo 2, algoritmos de agrupamento podem ser utilizados. Com a aplicação desses algoritmos, os pontos são agrupados em *clusters* e pontos isolados são considerados *outliers*. Em algoritmos baseados em densidade, como o algoritmo de Rodriguez e Laio (2014), *outliers* são os pontos que possuem baixa densidade (ρ) e alta distância para outros pontos com maior densidade (δ).

O presente trabalho se propôs a estudar a aplicação de algoritmos de agrupamento no espaço semântico, com o objetivo de detectar e remover *outliers* de documentos no âmbito de análise de sentimento na tentativa de melhorar o desempenho de uma CNN, em termos de taxa de acerto e desempenho computacional.

Para este fim, este trabalho propôs uma abordagem para resolução do problema de análise de sentimento, utilizando um algoritmo de agrupamento baseado em densidade para detecção de regiões de baixa densidade, de forma que palavras pertencentes a essas regiões sejam rotuladas como sendo *outliers*. Partindo da premissa de que essas palavras são pouco relevantes para a classificação de um documento, essas palavras são removidas dos documentos. Após a manipulação dos documentos, é utilizada uma CNN com uma camada de convolução para classificação. O modelo proposto é descrito a seguir.

4.1 DETECÇÃO DE REGIÕES DE BAIXA DENSIDADE NO ESPAÇO SEMÂNTICO

Em algoritmos de agrupamento baseados em densidade, os pontos são agrupados baseados em condições locais em uma única passada sobre a base de dados, permitindo a aplicação dessas técnicas em grandes bases de dados (Ester et al., 1996) (Hinneburg et al., 1998) (Rodriguez e Laio, 2014). Outra grande vantagem desse tipo de algoritmo é sua capacidade de detectar *clusters* de formato arbitrário.

No contexto desse trabalho, algoritmos de agrupamento baseados em densidade são ótimas opções para identificação de regiões de baixa densidade no espaço semântico em tempo de execução hábil, visto que a quantidade de pontos (palavras) pertencentes a um determinado problema pode ser muito grande, além da dificuldade em se determinar a distribuição dos dados previamente.

Conforme descrito na Seção 2.3, o algoritmo de agrupamento baseado em densidade de Rodriguez e Laio (2014) agrupa os pontos assumindo que centroides são pontos que contêm muitos vizinhos com menor densidade, e que esses centroides estão relativamente longe de outras áreas de alta densidade. Para encontrar esses centroides, o algoritmo calcula dois valores para todos os pontos: a densidade local ρ e a distância para outros pontos de maior densidade δ, segundo as Equações 2.7 e 2.9, respectivamente. Com esses valores, o algoritmo define que centroides são pontos com altos valores de ρ e δ. Consequentemente, *outliers* são pontos com baixos valores de ρ e altos valores de δ, ou seja, pontos com poucos vizinhos e distantes de regiões de alta densidade.

Esse algoritmo mostrou-se uma ótima opção para o contexto desse estudo, sendo capaz de trabalhar com somente uma passada sobre a base de dados, possuindo boa flexibilidade para ser usado em diferentes domínios de aplicação, além de apresentar bons resultados em problemas desafiadores com alta dimensionalidade em diferentes áreas, desde processamento de imagens (Rodriguez e Laio, 2014) até bioinformática (Wiwie et al., 2015).

Na primeira fase do modelo proposto, é executada uma adaptação do algoritmo de Rodriguez e Laio (2014), com o intuito de calcular os valores de ρ e δ para cada ponto. Para cálculo desses valores, é necessária uma medida de distância. Neste trabalho, foi utilizada a distância Euclidiana, apresentada na Equação 2.6, pela sua capacidade de capturar similaridade linguística e semântica entre palavras (Pennington et al., 2014). Com os valores de ρ e δ calculados para todos os pontos, é possível projetar esses dados em um gráfico, como exemplificado na Figura 4.1.

Em seguida, são definidos parâmetros ρ_{min} e δ_{max}, ou seja, um limiar mínimo para ρ e um limiar máximo para δ, respectivamente, de forma com que pontos com $\rho_i < \rho_{min}$ e $\delta_i > \delta_{max}$ serão considerados pontos pertencentes a regiões de baixa densidade. Esses pontos no espaço

semântico, que correspondem a palavras do vocabulário, são rotulados como *outliers*. A partir dos gráficos de $\rho \times \delta$, definem-se empiricamente os limiares de ρ_{min} e δ_{max}.

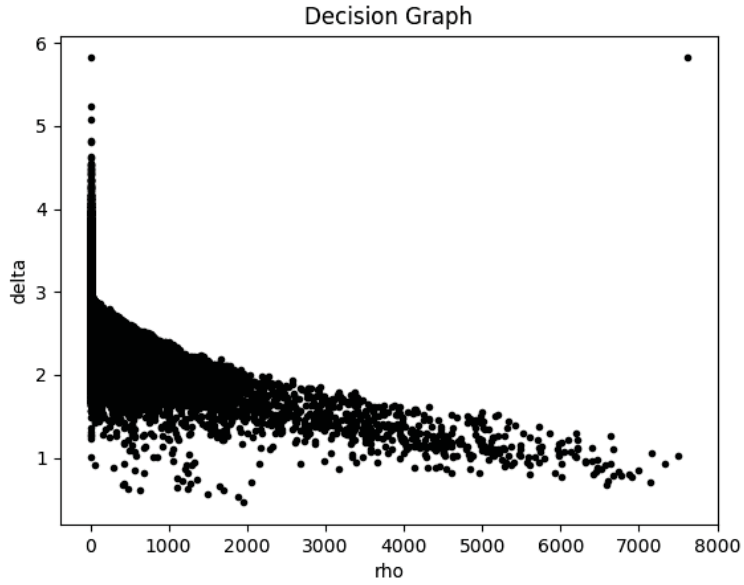


Figura 4.1: Gráfico da projeção rho × delta ($\rho \times \delta$). Esse gráfico foi obtido utilizando a representação *word2vec* com a base de dados MR.

Durante o processo de avaliação, cada documento é pré-processado antes do treinamento e classificação para garantir que os documentos não contenham nenhuma palavra rotulada como *outlier*. Após o pré-processamento dos documentos é efetuado o treinamento da CNN, cuja arquitetura é descrita a seguir.

4.2 ARQUITETURA

A arquitetura completa da CNN utilizada para treinamento, apresentada na Figura 4.2, é similar a arquitetura de Kim (2014), diferindo no número de canais de entrada, nos algoritmos utilizados para otimização e nas regularizações utilizadas, visto que o trabalho de Zhang e Wallace (2015) argumenta que alguns tipos de regularização têm pouco efeito no desempenho. Vale ressaltar que o modelo proposto pode ser aplicado em qualquer arquitetura, visto que a detecção de *outliers* é efetuada antes do treinamento do classificador.

Na primeira camada da rede, os documentos são *tokenizados*[1] e convertidos para uma matriz de frases, tal que cada linha da matriz se refere a uma palavra, e cada palavra é representada pelo seu vetor: neste trabalho, os vetores de palavras são inicializados com *word embeddings* pré-treinados, e essas representações são atualizadas (representações treináveis) durante o treinamento da rede.

Sendo d a dimensão dos vetores de palavra e s o tamanho de um dado documento, a matriz de frases M possui dimensões $s \times d$. Para que todos os documentos possuam mesmo tamanho, foi utilizada a técnica *zero-padding*, que consiste em representar todos os documentos com tamanho s pré-definido, sendo s o tamanho do maior documento da base de treinamento, e documentos inicialmente com tamanho menor a s são preenchidos com zeros.

[1]Procedimento de demarcação dos caracteres de entrada. Nesse caso, o documento é dividido em *tokens* (palavras) que são delimitados por um espaço em branco.

Com uma representação matricial dos documentos, é possível aplicar filtros de convolução. Todos os filtros aplicados possuem largura igual à dimensão dos vetores com o intuito de capturar toda a informação da palavra, variando-se somente sua altura, ou seja, o número de palavras a serem processadas juntas. Um filtro F de tamanho h aplicado sobre as palavras $M_{i:i+h-1}$ gera uma característica c_i dada pela Equação 4.1:

$$c_i = f(F \otimes M_{i:i+h-1} + b) \tag{4.1}$$

em que $\otimes$ representa uma operação de convolução, conforme definida na Equação 2.4, b representa o *bias* e f uma função de ativação. Nesse trabalho, foi utilizada a função *Rectified Linear Units* (ReLU), apresentada na Equação 4.2:

$$f(x) = max(0, x) \tag{4.2}$$

A aplicação do filtro F sobre todas as combinações de palavras no documento gera um vetor de características C dado pela Equação 4.3:

$$C = [c_1, c_2, ..., c_{s-h+1}] \tag{4.3}$$

em que s representa o tamanho do documento.

Em seguida é efetuada a operação de *max pooling* sobre o vetor de características C, que consiste em selecionar o maior valor c_F^{max}. Esse valor é considerado a característica mais importante do vetor, e é selecionado para representar o filtro F. A operação de *max pooling* é aplicável em documentos de tamanho arbitrário.

Na segunda camada da rede, são aplicadas diversas operações de convolução com diferentes filtros (e diferentes tamanhos de filtro) nas matrizes de frases, com o intuito de capturar várias características de uma mesma região, seguidos de operações de *max pooling*, conforme ilustrado na Figura 4.2. A concatenação de todos os valores de c_F^{max} compõem o vetor que é passado para a terceira camada da rede.

A terceira e última camada da rede é uma camada *softmax* totalmente conectada, responsável por atribuir a probabilidade do documento pertencer a uma determinada classe. Na terceira camada é aplicada regularização *dropout*, que consiste em aleatoriamente descartar (atribuir zero na saída) uma proporção p de nós ocultos durante o algoritmo *backpropagation*. Essa técnica é utilizada com o intuito de reduzir o *overfitting* (Srivastava et al., 2014).

4.2.1 Otimização

O modelo é treinado para minimizar a função objetivo entropia-cruzada, apresentada na Equação 4.4[2]:

$$H(p, q) = -\sum_x p_x \log(q_x) \tag{4.4}$$

em que x representa uma classe do problema, p representa a distribuição verdadeira (esperada), e q representa a distribuição estimada. Neste trabalho, a distribuição estimada q é obtida na última camada da rede, por meio da função *softmax*, definida na Equação 2.5.

Os gradientes são calculados com o algoritmo *backpropagation* (Rumelhart et al., 1986), e a atualização dos parâmetros da rede é feita com *Stochastic Gradient Descent* (SGD) com *minibatches* de tamanho pré-determinado, sendo esse um hiper-parâmetro da rede, por meio do algoritmo Adam (Kingma e Ba, 2014). A escolha desse algoritmo se dá pela sua boa capacidade

[2]Fonte: http://cs231n.github.io/linear-classify

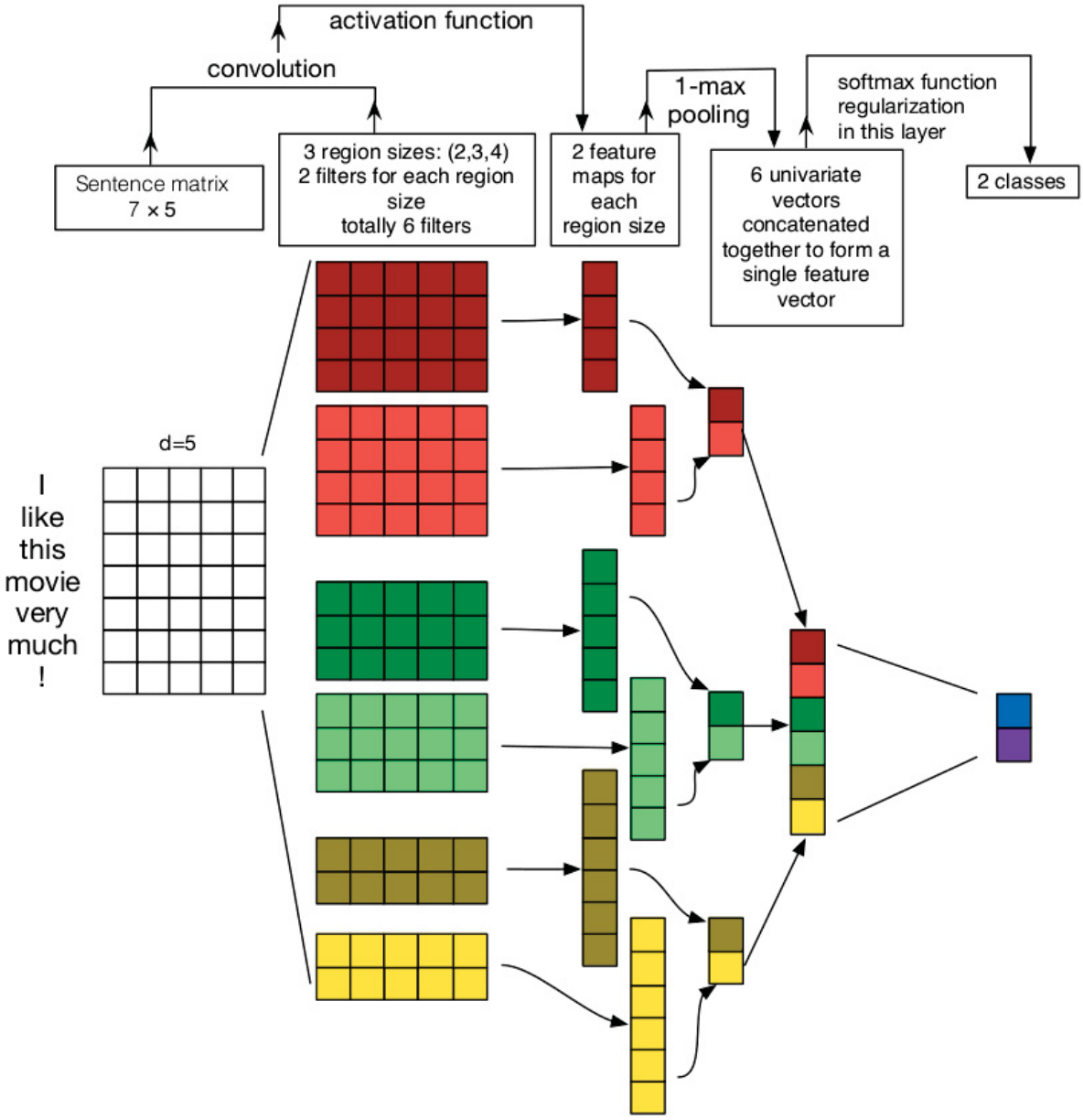


Figura 4.2: Ilustração de uma arquitetura de CNN para análise de sentimento. Nesse exemplo, são aplicados três tamanhos de filtros diferentes: 2, 3 e 4, com dois filtros para cada tamanho, tal que cada filtro é representado por uma cor diferente. A aplicação das convoluções gera vetores de características, e a função *max pooling* é aplicada sobre cada um deles. Os valores resultantes da operação de *pooling* são concatenados e passados para a camada *softmax*, que atribui probabilidades do documento pertencer a cada uma das classes (Zhang e Wallace, 2015).

de convergência e desempenho em comparação a outros algoritmos (Kingma e Ba, 2014) (Ruder, 2016). Os *minibatches* de treinamento são aleatoriamente embaralhados antes de cada época de treinamento com o intuito de melhorar a convergência (Bengio, 2012).

Os parâmetros atualizados durante o treinamento são: pesos dos filtros, *bias* para as funções não-lineares, vetores de pesos e vetores de palavras, seguindo o modelo treinável proposto por Kim (2014).

4.3 IMPLEMENTAÇÃO

Os algoritmos foram implementados na linguagem Python 2.7. A escolha se deu pela disponibilidade de bibliotecas e *frameworks* de aprendizado de máquina e NLP nessa linguagem (Bird et al., 2009) (Pedregosa et al., 2011) (Abadi et al., 2015). A implementação da CNN foi feita com a utilização do *framework Tensorflow* 1.4[3]. Esse *framework* foi escolhido pela sua flexibilidade, disponibilidade de diferentes operações e algoritmos voltados para *deep learning* e sua integração com GPUs.

Tensorflow é um *framework* para aprendizado de máquina que representa as computações de um algoritmo e o estado de seus componentes por meio de um grafo de fluxo de dados. Essa programação embasada em grafo de fluxo de dados permite a construção de diferentes arquiteturas com diferentes parâmetros e algoritmos de otimização sem que seja necessária a alteração da base do sistema principal, visto que as operações estão encapsuladas em nós do grafo (Abadi et al., 2016).

Tensorflow usa um único grafo de fluxo de dados que representa todas as operações e estados de um modelo de aprendizado de máquina, incluindo operações matemáticas, parâmetros do modelo e seus algoritmos de atualização, a entrada e pré-processamento, tal que uma aplicação de *Tensorflow* é definida como um grafo, usando *placeholders* para os dados de entrada e variáveis para armazenamento dos estados (Abadi et al., 2016). Na Figura 4.3 é apresentado o grafo de fluxo da arquitetura implementada.

Em um grafo de fluxo de dados do *Tensorflow*, cada vértice (nó) representa uma unidade de operação e cada aresta representa a saída de um nó e a entrada para outro nó. Os valores que fluem pelas arestas são chamadas de tensores. No *Tensorflow*, todos os dados são modelados como tensores, ou seja, matrizes n-dimensionais. Tensores representam entradas e resultados de operações matemáticas nas operações de algoritmos de aprendizado de máquina. Operações são definidas como procedimentos que recebem tensores como entrada e produzem tensores como saída. Uma operação pode conter estados mutáveis que são alterados durante o tempo de execução do grafo. Esses estados são determinados por meio de variáveis, que podem ser usadas para armazenar parâmetros compartilhados durante o treinamento (Abadi et al., 2016).

[3]https://www.tensorflow.org/

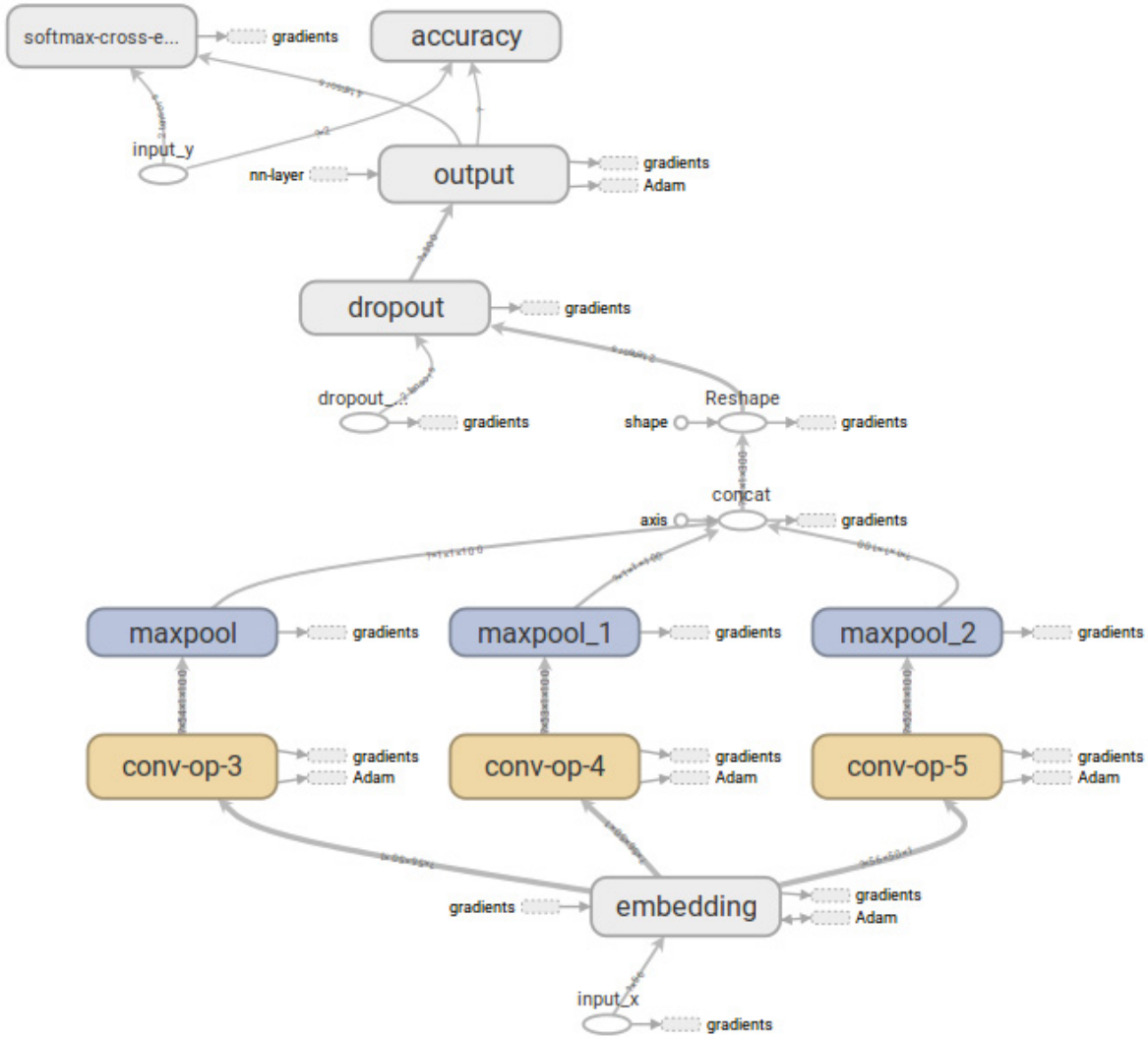


Figura 4.3: Grafo de fluxo de dados da arquitetura implementada. O grafo define o fluxo, as ligações e dependências entre as operações, começando pela entrada da rede (*input_x*), um *placeholder*. As arestas carregam tensores para outros nós, e sua orientação indica a dependência entre os nós. Cada operação é representada por um nó do grafo. Nesse grafo, as operações de convolução (*conv-op*-x) são representadas pelos tamanhos de filtro determinados pelos hiper-parâmetros do algoritmo (3, 4 e 5).

4.4 CONSIDERAÇÕES FINAIS

Este capítulo apresentou a abordagem proposta neste trabalho, que consiste na detecção de *outliers* no espaço semântico por meio da aplicação de um algoritmo de agrupamento baseado em densidade sobre as representações vetoriais de palavras. O modelo proposto é avaliado experimentalmente no próximo capítulo.

5 EXPERIMENTOS E RESULTADOS

Neste capítulo são apresentados os experimentos conduzidos para avaliação do modelo proposto no Capítulo 4 e os resultados obtidos. Inicialmente é abordada a metodologia adotada, incluindo as bases de dados utilizadas, hiper-parâmetros do modelo, representações de vetores de palavra utilizadas, detalhes da avaliação e informações do hardware utilizado. Em seguida, são apresentados experimentos referentes à descoberta de regiões de baixa densidade. Depois, são apresentados os resultados dos experimentos obtidos, incluindo análises desses resultados. Por fim, uma síntese dos resultados obtidos é apresentada ao final do capítulo.

5.1 METODOLOGIA

Para avaliar o modelo proposto, foi utilizada como base de comparação a arquitetura da CNN sem a aplicação do algoritmo de agrupamento e a remoção de *outliers*. O modelo foi avaliado em bases de dados de análise de sentimento, com dois algoritmos de obtenção de vetores de palavras.

Para bases de dados sem conjunto de testes definidos, foi utilizada validação cruzada com 10-*folds*. Experimentos sobre essas bases de dados foram executados 10 vezes, a fim de medir a significância estatística dos resultados. Em bases de dados com conjunto de testes definidos previamente, os experimentos foram executados uma vez, e depois executados 5 vezes com validação cruzada com 2-*folds*, combinando as bases de treinamento e teste, para significância estatística. Em todos os experimentos, as palavras fora de vocabulário são descartadas.

Os primeiros experimentos consistiram na determinação das regiões de baixa densidade, para obter valores adequados para os limiares de ρ_{min} e δ_{max}. Após a obtenção dos valores limiares, foram executados experimentos para avaliar o modelo proposto, analisando o impacto da remoção de *outliers*. Os parâmetros dos experimentos, resultados e discussões são apresentados a seguir.

5.1.1 Validação Cruzada

A validação cruzada *k-folds* consiste em particionar aleatoriamente o conjunto de dados original em k subconjuntos de mesmo tamanho. Em uma iteração do processo de validação cruzada, seleciona-se um dos k subconjuntos como base de validação para testar o modelo, e os demais $k - 1$ subconjuntos são utilizados como dados de treinamento. Ao final da iteração, calcula-se a taxa de acerto obtida sobre a base de validação. Esse procedimento é repetido k vezes (número de *folds*), tal que cada subconjunto é utilizado uma vez como base de validação. Ao final das k iterações, calcula-se a média das taxas de acerto obtidas para produzir uma estimativa do modelo sobre a base original (Kohavi et al., 1995).

A validação cruzada *k-folds* garante que todas as amostras são utilizadas para treino e validação, tal que cada amostra é utilizada para validação somente uma vez. A validação cruzada 10-*folds* é recomendada, por conta de sua reduzida variância (Kohavi et al., 1995).

5.1.2 Bases de Dados

Para avaliar o modelo proposto, foram executados experimentos usando cinco bases de dados de análise de sentimento. As bases de dados utilizadas foram escolhidas por pertencerem a diferentes domínios de aplicação e por possuírem diferentes tamanhos de documento, além de serem utilizadas em diversos trabalhos. Informações estatísticas referentes às bases de dados são apresentadas na Tabela 5.1. As bases de dados são descritas brevemente a seguir:

- MR (*Movie reviews dataset*): base de dados de análises de filmes. Cada documento (análise) contém uma frase, que expressa uma opinião positiva ou negativa (Pang e Lee, 2005)[1];
- Subj (*Subjectivity dataset*): base de dados de subjetividade, tal que cada documento contém uma frase que pode ser classificada como objetiva, expressando fatos, ou subjetiva, expressando a opinião do autor. As frases foram extraídas de análises de filmes (documentos subjetivos) e de resumos de filmes (documentos objetivos) por Pang e Lee (2005)[1];
- IMDB: base de dados de filmes, em que cada documento representa uma análise completa de um filme, e cada documento deve ser classificado como positivo ou negativo (Maas et al., 2011)[2];
- Elec (*Electronic product reviews*): base de dados de produtos eletrônicos, em que cada documento contém uma análise de um produto eletrônico, e cada documento deve ser classificado como positivo ou negativo (McAuley e Leskovec, 2013). A versão utilizada foi adaptada e redistribuída por Johnson e Zhang (2015a)[3];
- Yelp (*Yelp Review Polarity*): base de dados de avaliações de diferentes tipos de serviços, obtida por meio do *Yelp Dataset Challenge*[4] em 2015, que contém 1.569.264 exemplos. A partir dessa base, em Zhang et al. (2015) foram construídas duas bases de dados, sendo uma delas para prever a polaridade de um documento, em que documentos com 1 e 2 estrelas de avaliação são considerados negativos e documentos com 3 e 4 estrelas são considerados positivos. Essa base de dados de polaridade contém 560.000 dados para treinamento, e 38.000 dados para testes, sendo os dados distribuídos igualmente entre as polaridades, mantendo a proporção. Por conta do tamanho dessa base de dados, foram selecionados 10% dos dados de treinamento e 10% dos dados de testes para compor a base de dados utilizada nos experimentos deste trabalho. A proporção da polaridade dos dados foi mantida.

5.1.3 *Word embeddings*

Para aplicação do algoritmo de agrupamento e treinamento da CNN, foram utilizados dois algoritmos de obtenção de *word embeddings*: *gloVe* e *word2vec*, conforme definidos na Seção 2.1.2.

O algoritmo *gloVe* faz o treinamento dos vetores sobre estatísticas de co-ocorrência de palavras extraídas de um corpus. As representações pré-treinadas utilizadas, disponibilizadas

[1] https://www.cs.cornell.edu/people/pabo/movie-review-data/
[2] http://www.andrew-maas.net/data/sentiment
[3] http://riejohnson.com/cnn_data.html
[4] http://www.yelp.com/dataset_challenge

Tabela 5.1: Informações sobre as bases de dados. T_{avg}: quantidade média de palavras nos documentos. T_{max}: quantidade de palavras no maior documento. N: tamanho da base de dados. V: quantidade de palavras no vocabulário após pré-processamento. V_{w2v}: quantidade de palavras do vocabulário existentes no *word2vec* pré-treinado. V_{gloVe}: quantidade de palavras do vocabulário existentes no *gloVe* pré-treinado. Teste: tamanho da base de testes; em bases de dados sem base de testes definida, foi aplicada validação cruzada 10-*folds* (VC).

Base de dados	T_{avg}	T_{max}	N	V	V_{w2v}	V_{gloVe}	Teste
MR	20	56	10662	18337	16475	17634	VC
Subj	23	120	10000	20909	17950	20031	VC
IMDB	251	2633	50000	105876	58955	75452	25000
Elec	115	5715	50000	65711	33644	35180	25000
Yelp	144	1080	59800	71329	44605	48233	3800

publicamente[5], foram treinadas utilizando um corpus extraído das bases *Wikipedia2014* e *Gigaword5*, com vocabulário de 400.000 de palavras, sendo 50 a dimensionalidade dos vetores de palavras (Pennington et al., 2014).

A ferramenta *word2vec* disponibiliza implementações eficientes das arquiteturas CBOW e *skip-gram* para obtenção de representações de palavras. A ferramenta tem como entrada um corpus de texto e produz os vetores de palavras como saída. As representações pré-treinadas utilizadas, disponibilizadas publicamente[6], foram treinadas com a arquitetura CBOW, utilizando parte da base *Google News* com vocabulário de 3 milhões de palavras, sendo 300 a dimensionalidade dos vetores de palavras (Mikolov et al., 2013a).

5.1.4 Hiper-parâmetros

Os hiper-parâmetros da CNN foram selecionados empiricamente considerando os resultados obtidos por Kim (2014) e Zhang e Wallace (2015) e resultados de experimentos preliminares sobre a base de dados MR. Foram utilizadas duas configurações de parâmetros, sendo a diferença entre elas o modelo de representação de palavras utilizado (algoritmo de obtenção de *word embeddings*) e as dimensões dos vetores de palavras. As configurações de parâmetros utilizadas nos experimentos são apresentadas na Tabela 5.2.

5.1.5 Hardware

Os experimentos foram executados em uma máquina com sistema operacional Ubuntu 15.10, com a seguinte especificação:

- 16 GB de memória RAM;
- Processador Intel Core i7-4770 com frequência de 3.4 GHz;
- GPU NVIDIA 960 com 4 GB.

5.1.6 Regiões de Baixa Densidade

Os primeiros experimentos foram executados com o intuito de determinar as regiões de baixa densidade para ambas as representações utilizadas, ou seja, regiões que contêm palavras

[5]https://nlp.stanford.edu/projects/glove/

[6]https://code.google.com/archive/p/word2vec/

Tabela 5.2: Configurações de hiper-parâmetros da CNN utilizadas nos experimentos. *Word embedding*: representação dos vetores de palavra. *Dim*: Dimensão dos vetores de palavras. $F_$size: Tamanhos de filtros de convolução utilizados. N_{F_size}: Quantidade de filtros de convolução por tamanho de filtro. *Dropout*: taxa da regularização *dropout*. *Batch*: Tamanho dos *minibatches* usados no treinamento. Épocas: épocas de treinamento utilizadas. *Shuffle*: indica se os *minibatches* são re-embaralhados durante cada época de treinamento.

Hiper-parâmetro	Configuração 1	Configuração 2
Word embedding	*word2vec*	*gloVe*
Dim	300	50
$F_$size	(3, 4, 5)	(3, 4, 5)
N_{F_size}	100	100
Dropout	0,5	0,5
Batch	50	50
Épocas	25	25
Shuffle	*true*	*true*

que não são similares a outras palavras no espaço semântico, e consequentemente podem ser consideradas como sendo *outliers*. Para esses experimentos, a base de dados MR foi selecionada como base de definição. O algoritmo de agrupamento baseado em densidade foi executado, obtendo os valores de ρ e δ para todos os pontos, permitindo a execução de experimentos com diferentes limiares de ρ_{min} e δ_{max}. As projeções de $\rho \times \delta$ de todos os pontos para as representações *word2vec* e *gloVe* são apresentadas na Figura 5.1. Os *outliers* obtidos com os diferentes parâmetros foram removidos dos vocabulários, e a CNN foi treinada com esses vocabulários para verificar a variação na taxa de acerto do modelo.

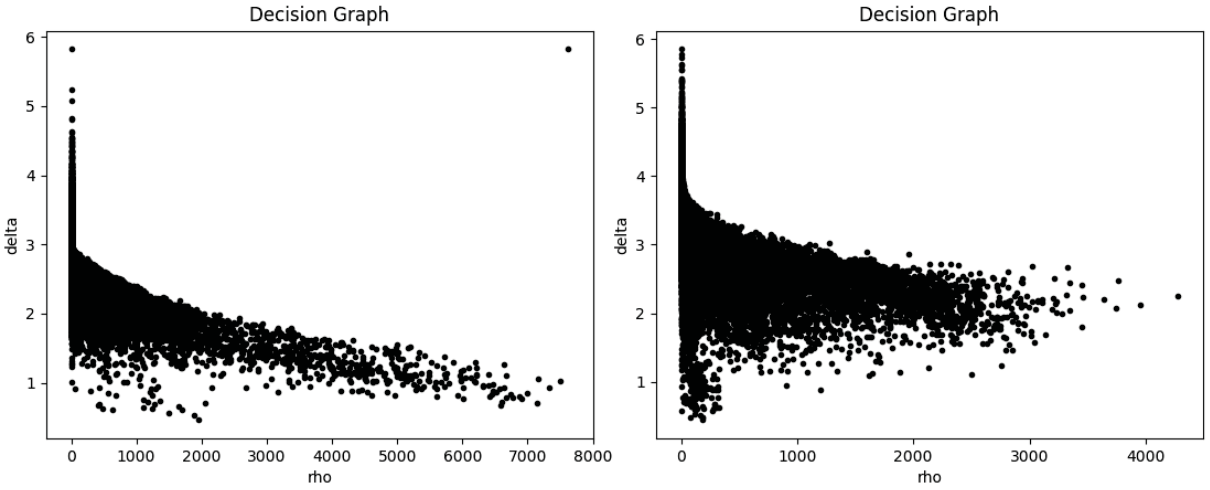


Figura 5.1: Gráficos das projeções dos valores de $\rho \times \delta$ para todos os pontos da base MR com as representações *word2vec* e *gloVe*, respectivamente.

Pela definição do algoritmo de agrupamento baseado em densidade utilizado, para cada ponto i, ρ_i mede a densidade local, enquanto δ_i representa a distância mínima entre o ponto e qualquer outro ponto com maior densidade, ou seja, no espaço semântico, um valor baixo de ρ significa que o ponto possui poucos vizinhos, e um valor alto de δ indica que o ponto está semanticamente longe de outros pontos. Esses pontos são considerados como *outliers*. A fim de encontrar esses *outliers*, foram executados experimentos variando os parâmetros ρ_{min} e δ_{max}.

Primeiro, foi testada a variação causada pelo parâmetro ρ_{min}, com δ_{max} fixado em 0 com o intuito de verificar o impacto do limiar de densidade local. Esses experimentos são apresentados nas Tabelas 5.3 e 5.4 para as representações *word2vec* e *gloVe*, respectivamente.

Tabela 5.3: Experimentos para determinação das regiões de baixa densidade da representação *word2vec* com o parâmetro $\delta_{max} = 0$.

Limiar ρ_{min}	**Taxa acerto (%)**	**Desvio padrão**	**Número *Outliers***	**Tempo (s)**
-	81,33	1,16	-	1682,69
5	79,94	1,31	6372	1290,63
10	79,33	1,31	7429	1220,45
15	78,95	1,16	7989	1186,74
20	78,52	1,32	8364	1164,83
25	78,29	1,36	8662	1149,41
50	77,18	1,23	9564	1069,89
100	76,13	1,33	10578	1006,21
250	74,09	1,34	12144	856,01
500	70,99	1,39	13624	744,19
1000	67,01	1,31	14998	631,89
1500	64,64	1,46	15542	430,39
2000	63,44	0,63	15820	407,88

Tabela 5.4: Experimentos para determinação das regiões de baixa densidade da representação *gloVe* com o parâmetro $\delta_{max} = 0$.

Limiar ρ_{min}	**Taxa acerto (%)**	**Desvio padrão**	**Número *Outliers***	**Tempo (s)**
-	77,1	1,25	-	450,13
5	77,31	1,3	2512	373,51
10	77,31	1,38	3656	358,48
15	77,07	1,31	4420	354,32
20	76,79	1,34	5064	348,47
25	76,71	1,54	5598	339,81
50	76,43	1,28	7283	324,8
100	75,58	1,36	9023	301,83
250	73,67	1,41	11556	270,27
500	70,46	1,37	13624	231,56
1000	62,68	1,42	15600	159,85
1500	57,82	1,73	16531	140,78
2000	55,42	1,57	17110	133,05

Os resultados apresentados nas Tabelas 5.3 e 5.4 foram projetados em gráficos que podem ser encontrados na Figura 5.2. Observando esses resultados, pode-se verificar que para o

word2vec, a taxa de acerto cai para qualquer valor de ρ_{min} com $\delta_{max} = 0$. Observa-se também que a taxa de acerto diminui conforme aumenta-se o valor de ρ_{min}, ou seja, mais palavras são consideradas como sendo *outliers*. Embora haja queda na taxa de acerto, podem ser removidas muitas palavras do vocabulário, sem causar quedas significativas. A partir de $\rho_{min} = 100$ a queda na taxa de acerto ultrapassa 5%, no entanto, são removidas 10578 palavras do vocabulário, o que representa 64,2% do vocabulário original: esse vocabulário reduzido reflete no tempo de treinamento da rede, que é reduzido em torno de 40%. Essa queda no tempo de processamento é observada em todos os valores de ρ_{min}: isso se dá pelo reduzido tamanho do vocabulário e pelo menor tamanho dos documentos de entrada, o que diminui a dimensionalidade dos vetores de entrada e também da matriz de palavras, que é um parâmetro treinável da CNN.

Para a representação *gloVe*, a taxa de acerto aumentou para os valores de $\rho_{min} = 5$ e $\rho_{min} = 10$, e o tempo de execução da CNN diminuiu. A partir de $\rho_{min} = 15$, a taxa de acerto diminui conforme aumenta-se o valor de ρ_{min}. Usando essa representação, a remoção de 9023 palavras do vocabulário ($\rho_{min} = 100$) causou uma queda de 1,52% na taxa de acerto original. A partir de $\rho_{min} = 250$, a queda na taxa de acerto torna-se mais significativa, conforme pode-se observar na Figura 5.2. Assim como no *word2vec*, a utilização do *gloVe* também apresentou queda no tempo de execução da CNN, embora essa queda seja menor que a obtida com o *word2vec*, por conta da menor dimensionalidade dos vetores de palavras.

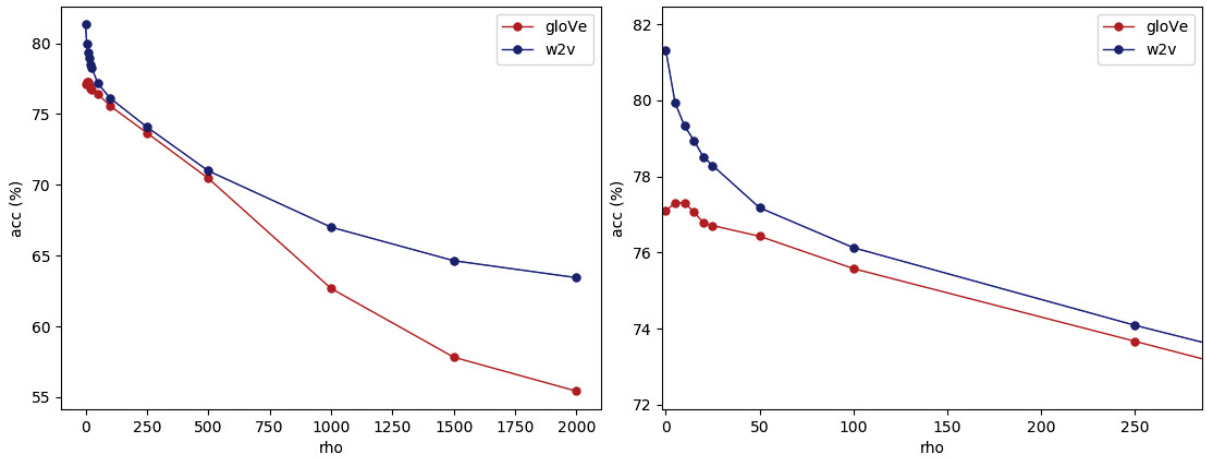


Figura 5.2: Gráficos da projeção $\rho_{min} \times acc$ (taxa de acerto) dos resultados apresentados nas Tabelas 5.3 e 5.4. O gráfico da esquerda representa todos os pontos da tabela, enquanto o gráfico da direita representa os pontos com valores de ρ_{min} entre 0 e 250.

Com base nesses resultados, definiram-se 5, 10 e 15, como valores de ρ_{min} de interesse para o *word2vec* e 5, 10, 15, 20, 25 e 50 como valores de interesse para o *gloVe* para experimentos subsequentes. Esses valores foram escolhidos devido à alteração relativamente baixa nas taxas de acerto.

O próximo conjunto de experimentos foi executado para verificar o efeito do limiar de δ_{max}. Foram utilizadas combinações diferentes de valores para ρ_{min} e δ_{max} para as duas representações por conta da diferença de comportamento entre os pontos, conforme observado na Figura 5.1. Esses resultados são apresentados nas Tabelas 5.5 e 5.6 para o *word2vec* e *gloVe*, respectivamente, e projetados em mapas de calor apresentados na Figura 5.3.

Com base nesses resultados, para a representação *word2vec*, pode-se verificar que a taxa de acerto aumenta conforme o limiar de δ_{max} cresce, até $\delta_{max} = 2{,}5$. Considerando $\rho_{min} = 15$, a taxa de acerto cresce até $\delta_{max} = 3$, e diminui quando $\delta_{max} = 3{,}5$, no entanto, a taxa de acerto

Tabela 5.5: Experimentos para determinação das regiões de baixa densidade da representação *word2vec*, variando-se o parâmetro δ_{max} com ρ_{min} fixado em 5, 10 e 15. Foram executados testes com $\delta_{max} > 3$ somente para $\rho_{min} = 15$ porque o número de *outliers* é o mesmo para todos os valores de $\rho_{min} < 15$.

Limiar δ_{max}	**Limiar** ρ_{min}	**Taxa acerto (%)**	**Desvio padrão**	**Número *Outliers***	**Tempo (s)**
-	-	81,33	1,16	-	1682,69
1.5	1	79,58	1,18	4051	1431,21
2.0	1	79,62	1,12	4030	1434,51
2.5	1	79,82	1,13	3780	1447,01
1.5	2	79,18	1,36	4933	1373,51
2.0	2	79,14	1,4	4894	1370,46
2.5	2	79,51	1,29	4489	1387,29
1.5	3	78,84	1,09	5548	1303,04
2.0	3	78,78	1,22	5499	1314,64
2.5	3	79,11	1,29	4932	1351,47
1.5	4	78,47	1,15	6013	1302,37
2.0	4	78,53	1,19	5937	1291,23
2.5	4	79,02	1,38	5261	1364,06
1.5	5	78,46	1,32	6372	1252,69
2.0	5	80,16	1,41	6280	1271,63
2.5	5	80,61	1,27	5499	1337
1.5	10	79,64	1,39	7426	1245,81
2.0	10	79,68	1,48	7285	1244,81
2.5	10	80,22	1,48	6183	1297,6
1.5	15	79,31	1,21	7983	1191,68
2.0	15	79,36	1,38	7818	1202,46
2.5	15	79,81	1,41	6529	1279,32
3.0	15	81,5	1,17	2851	1522,31
3.5	15	81,4	1,17	450	1668,85

ainda foi maior do que a taxa de acerto obtida sem a remoção de *outliers*. Considerando-se os resultados com δ_{max} até 2,5, os melhores resultados foram obtidos com $\rho_{min} = 5$.

Para a representação *gloVe*, a taxa de acerto aumentou, mesmo que ligeiramente, para todos valores de ρ_{min} com $\delta_{max} = 3{,}5$. Para valores de ρ_{min} entre 5 e 50, a taxa de acerto cresce considerando δ_{max} entre 2,5 e 3. Considerando valores de ρ_{min} entre 1 e 10, a taxa de acerto caiu com $\delta_{max} = 3{,}5$ comparado com $\delta_{max} = 3$. O melhor resultado foi obtido com $\rho_{min} = 10$ e $\delta_{max} = 3$. Como a taxa de acerto foi melhor com $\delta_{max} = 3{,}5$ do que $\delta_{max} = 2{,}5$ para $\rho_{min} = 10$, foi definida como região de interesse para o *gloVe* os valores de δ_{max} entre 3 e 3,5.

Baseado nessas análises, para a representação *word2vec* foram selecionados como valores ideais para δ_{max} o intervalo entre 2,5 e 3,5. Como os melhores resultados para $\delta_{max} = 2{,}5$ foram obtidos com $\rho_{min} = 5$, o valor ideal para ρ_{min} foi fixado em 5. Para a representação *gloVe* foram escolhidos como valores ideais para δ_{max} o intervalo entre 3 e 3,5. Como os melhores resultados foram obtidos com $\rho_{min} = 10$, o valor ideal para ρ_{min} foi fixado em 10. Foi utilizado um intervalo maior com a representação *word2vec* para observar a partir de qual limiar de δ_{max} a taxa de acerto supera a taxa de acerto sem remoção de *outliers*.

Tabela 5.6: Experimentos para determinação das regiões de baixa densidade da representação *gloVe*, variando-se o parâmetro δ_{max} com ρ_{min} fixado em 5, 10, 15, 20, 25 e 50. Foram executados testes com $\delta_{max} > 4$ somente para $\rho_{min} = 50$ porque o número de *outliers* é o mesmo para todos os valores de $\rho_{min} < 50$.

Limiar δ_{max}	**Limiar** ρ_{min}	**Taxa acerto (%)**	**Desvio padrão**	**Número *Outliers***	**Tempo (s)**
-	-	77,1	1,25	-	450,13
3.0	1	77,36	1,26	1133	396,72
3.5	1	77,22	1,3	1088	395,88
3.0	2	77,27	1,34	1503	398,59
3.5	2	77,21	1,31	1381	392,59
3.0	3	77,36	1,35	1847	390,83
3.5	3	77,27	1,48	1641	393,86
3.0	4	77,31	1,3	2120	387,31
3.5	4	77,23	1,34	1827	392,85
2.0	5	76,26	1,19	2500	381,43
2.5	5	77,23	1,37	2467	383,82
3.0	5	77,4	1,38	2339	386,65
3.5	5	77,27	1,29	1976	390,02
2.0	10	75,89	1,33	3628	372,27
2.5	10	77,3	1,26	3557	369,56
3.0	10	77,49	1,32	3300	377,76
3.5	10	77,39	1,34	2539	383,92
2.0	15	75,46	1,42	4381	362,37
2.5	15	75,58	1,34	4267	362,99
3.0	15	75,69	1,37	3889	371,28
3.5	15	77,2	1,24	2801	378,09
2.0	20	75,38	1,26	5018	355,13
2.5	20	75,5	1,28	4866	357,09
3.0	20	75,62	1,27	4358	365,73
3.5	20	77,17	1,26	2983	375,42
2.0	25	75,28	1,31	5541	350,62
2.5	25	75,33	1,3	5361	353,22
3.0	25	75,47	1,14	4732	361,74
3.5	25	77,21	1,25	3109	379,54
2.0	50	73,7	1,34	7182	336,69
2.5	50	73,96	1,26	6875	339,33
3.0	50	74,9	1,24	5842	351,09
3.5	50	77,14	1,3	3358	373,59
4.0	50	77,2	1,22	834	397,86

5.2 RESULTADOS

Com os valores ideais de ρ_{min} e δ_{max} definidos para as duas representações, foram executados experimentos com cinco bases de dados, a fim de avaliar o impacto da remoção de *outliers* no desempenho da rede em termos de taxa de acerto e tempo de processamento. Os

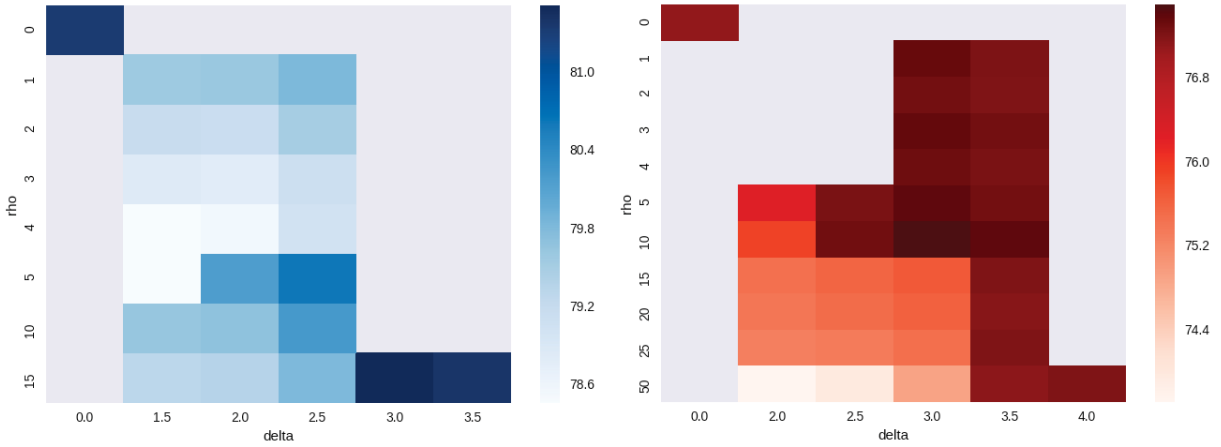


Figura 5.3: Mapas de calor dos resultados apresentados nas Tabelas 5.5 e 5.6. O mapa da esquerda apresenta os resultados da representação *word2vec*, enquanto o mapa da direita apresenta os resultados da representação *gloVe*. Em ambos os mapas, a taxa de acerto é representada pelas cores, tal que cores mais escuras representam valores maiores de taxa de acerto.

experimentos seguem a metodologia descrita na subseção 5.1, e são comparados com resultados da arquitetura sem a remoção de *outliers*. Para a representação *word2vec*, os resultados são apresentados nas Tabelas 5.7 e 5.8, e para a representação *gloVe* os resultados são apresentados nas Tabelas 5.9 e 5.10.

Tabela 5.7: Resultados para a representação *word2vec* com remoção de *outliers* para as bases de dados MR e Subj. Células em destaque indicam os melhores resultados obtidos em termos de taxa de acerto. Legenda: acc (%): taxa de acerto; *stdev*: desvio padrão; *Outliers*: quantidade de palavras consideradas como *outliers* com base no valor dos limiares; Tempo (s): tempo total de treinamento e validação da CNN em segundos.

	MR				**Subj**			
δ_{max}	**acc** (%)	**stdev**	***Outliers***	**Tempo** (s)	**acc** (%)	**stdev**	***Outliers***	**Tempo** (s)
-	81,33	1,16	-	1683,33	93,17	0,751	-	2000,48
2.5	80,61	1,27	5499	1326,51	92,59	0,832	6925	1643,93
2.6	80,81	1,23	5227	1379,93	92,68	0,864	6500	1650,09
2.7	80,97	1,19	4857	1388,01	92,77	0,796	5945	1661,75
2.8	81,09	1,22	4394	1414,35	93	0,743	5196	1683,67
2.9	81,15	1,09	3745	1463,79	93	0,787	4198	1716,71
3.0	81,5	1,17	2849	1519,76	93,29	0,711	3125	1730,63
3.1	**81,77**	**1,05**	**2010**	**1573,17**	93,41	0,776	2261	1816,89
3.2	81,69	1,08	1445	1585	**93,43**	**0,731**	**1632**	**1889,99**
3.3	81,66	1,13	994	1616,91	93,36	0,76	1150	1900,87
3.4	81,66	1,13	666	1647,73	93,36	0,745	787	1932,66
3.5	81,4	1,03	450	1668,85	93,21	0,787	516	1944,95

Analisando os resultados apresentados nas Tabelas 5.7, 5.8, 5.9 e 5.10, é possível observar que a remoção de *outliers* impacta ligeiramente no desempenho da arquitetura, aumentando a taxa de acerto e reduzindo o tempo de treinamento para as duas representações utilizadas sobre todas as bases de dados. Os melhores resultados em termos de taxa de acerto foram obtidos com

Tabela 5.8: Resultados para a representação *word2vec* com remoção de *outliers* para as bases de dados IMDB, Elec e Yelp. Células em destaque indicam os melhores resultados obtidos em termos de taxa de acerto. Legenda: acc (%): taxa de acerto; *Outliers*: quantidade de palavras consideradas como *outliers* com base no valor dos limiares; Tempo (s): tempo total de treinamento e validação da CNN em segundos.

	IMDB			**Elec**			**Yelp**		
δ_{max}	**acc** (%)	***Outliers***	**Tempo** (s)	**acc** (%)	***Outliers***	**Tempo** (s)	**acc** (%)	***Outliers***	**Tempo** (s)
-	90,06	-	9135,1	90,27	-	16201,8	93,68	-	8059,7
2.5	89,46	32081	6143,1	90,0	16980	13227,2	92,58	23090	5977,8
2.6	89,9	28017	6742,0	90,42	14911	13978,5	93,26	20249	6548,7
2.7	90,08	23880	7060,5	90,51	12853	13753,5	93,45	17316	6774,7
2.8	**90,37**	**19720**	**7068,1**	90,53	10712	14752,9	93,65	14420	7126,7
2.9	90,29	15658	7270,2	**90,55**	**8550**	**14929,7**	93,78	11568	7286,1
3.0	90,28	12200	7714,9	90,49	6714	14383,3	93,82	9092	7447,8
3.1	90,25	9199	7934,8	90,52	5117	14529,0	**94,0**	**6894**	**7555,3**
3.2	90,31	6814	8170,3	90,46	3798	14635,1	93,93	5152	7671,9
3.3	90,27	4944	8290,8	90,47	2726	14717,7	93,88	3752	7749,8
3.4	90,27	3431	8507,1	90,47	1948	15556,8	93,83	2675	7842,1
3.5	90,29	2394	8776,8	90,45	1361	15610,9	93,8	1886	7848,3

Tabela 5.9: Resultados para a representação *gloVe* com remoção de *outliers* para as bases de dados MR e Subj. Células em destaque indicam os melhores resultados obtidos em termos de taxa de acerto. Legenda: acc (%): taxa de acerto; *stdev*: desvio padrão; *Outliers*: quantidade de palavras consideradas como *outliers* com base no valor dos limiares; Tempo (s): tempo total de treinamento e validação da CNN em segundos.

	MR				**Subj**			
δ_{max}	**acc** (%)	**stdev**	***Outliers***	**Tempo** (s)	**acc** (%)	**stdev**	***Outliers***	**Tempo** (s)
-	77,1	1,25	-	450,13	91,96	0,904	-	542,95
3.0	77,49	1,32	3300	377,76	92,19	0,822	2822	499,9
3.1	77,27	1,28	3213	382,76	**92,23**	**0,921**	**2741**	**502,14**
3.2	**77,58**	**1,23**	**3105**	**385,33**	92,19	0,944	2646	503,95
3.3	77,44	1,32	2948	385,81	92,22	0,839	2535	505,17
3.4	77,31	1,33	2764	387,69	92,22	0,874	2433	506,79
3.5	77,39	1,34	2539	393,89	92,2	0,79	2306	510,72

o *word2vec*, embora essa maior taxa de acerto possa ser justificada pelas dimensões dos vetores de palavras (300 para o *word2vec* e 50 para o *gloVe*).

Também é possível notar que o tempo de execução diminui conforme a quantidade de *outliers* eliminados aumenta, visto que o tamanho do vocabulário também diminui. Considerando o *word2vec* e a base de dados IMDB, com δ_{max} = 2,5 o tempo de execução é reduzido em mais de 30%. Para o *gloVe*, embora haja uma queda no tempo de processamento, essa queda é baixa, o que também pode ser atribuído ao tamanho dos vetores de palavras: quanto maior a dimensão dos vetores de palavras, maior a quantidade de parâmetros a serem treinados pela rede. Assim,

Tabela 5.10: Resultados para a representação *gloVe* com remoção de *outliers* para as bases de dados IMDB, Elec e Yelp. Células em destaque indicam os melhores resultados obtidos em termos de taxa de acerto. Legenda: acc (%): taxa de acerto; *Outliers*: quantidade de palavras consideradas como *outliers* com base no valor dos limiares; Tempo (s): tempo total de treinamento e validação da CNN em segundos.

	IMDB			**Elec**			**Yelp**		
δ_{max}	**acc** (%)	***Outliers***	**Tempo** (s)	**acc** (%)	***Outliers***	**Tempo** (s)	**acc** (%)	***Outliers***	**Tempo** (s)
-	87,29	-	2425,8	89,13	-	4718,2	92,07	-	2145,1
3.0	87,65	11468	2323,9	**89,44**	**4034**	**4434,6**	92,44	5483	2052,1
3.1	**87,66**	**11184**	**2339,2**	**89,44**	**3941**	**4434,9**	92,41	5368	2052,4
3.2	87,63	10834	2324,8	89,43	3857	4439,8	92,4	5243	2058,5
3.3	87,57	10422	2340,6	89,4	3734	4440,8	**92,5**	**5100**	**2065,1**
3.4	87,5	9896	2342,2	89,39	3609	4446,2	92,35	4917	2064,1
3.5	87,57	9176	2359,2	89,37	3441	4455,0	92,33	4676	2067,5

a escolha de limiares de ρ_{min} e δ_{max} pode ser um *trade-off* entre taxa de acerto e tempo de execução.

Para a representação *word2vec*, considerando as bases MR e Subj, a taxa de acerto melhora em relação à arquitetura sem remoção de *outliers* a partir de δ_{max} = 3, sendo os melhores resultados com δ_{max} = 3,1 e δ_{max} = 3,2 respectivamente. Para as demais bases de dados, a taxa de acerto da arquitetura melhora a partir dos limiares de δ_{max}: 2,7 para a base IMDB, 2,6 para a base Elec e 2,9 para a base Yelp. Com esses limiares, são removidas muitas palavras do vocabulário sem causar impacto negativo na taxa de acerto, o que é possível devido ao grande tamanho médio dos documentos, que são compostos de análises completas, diferente das bases de dados MR e Subj que são compostas por frases curtas.

Para a representação *gloVe*, os melhores resultados foram obtidos com o intervalo de δ_{max} entre 3 e 3,3. Para todas as bases de dados, foram obtidos resultados melhores do que a arquitetura sem remoção de *outliers*, independentemente do valor do limiar de δ_{max}. Para o *gloVe*, a variação na taxa de acerto entre os limiares de δ_{max} é muito pequena por conta da pequena diferença entre a quantidade de *outliers* removidos entre os limiares.

Os valores ideais de δ_{max} foram em sua maior parte muito próximos para as duas representações. Além disso, vale ressaltar que para a representação *word2vec*, a quantidade de *outliers* removidos entre limiares de δ_{max} é consideravelmente maior do que com a representação *gloVe*. No entanto, a variação na taxa de acerto é pequena, mesmo com muitas palavras sendo consideradas *outliers*, o que indica que a dimensão dos vetores de palavras pode ter grande impacto na taxa de acerto de uma arquitetura.

5.2.1 Significância Estatística

Para avaliar a significância estatística dos resultados, foram consideradas as 10 execuções da validação cruzada 10-*folds* (10x10-*folds*) para as bases de dados MR e Subj. Para as bases de dados IMDB, Elec e Yelp, as bases de treinamento e testes foram combinadas e foram conduzidos experimentos 5x2-*folds*, ou seja, 5 execuções de validação cruzada 2-*folds*.

Para comparação entre dois algoritmos, a validação cruzada 10x10-*folds* é recomendada por conta de seu baixo erro do tipo I (falso positivo) (Bouckaert, 2003). Em situações em que a execução de validação cruzada 10x10-*folds* é inviável, recomenda-se a execução de validação

cruzada 5x2-*folds* (Dietterich, 1998) (Bouckaert, 2003). Para as bases de dados IMDB, Elec e Yelp, os experimentos foram conduzidos utilizando os hiper-parâmetros da Tabela 5.2, no entanto, foram utilizadas 10 épocas de treinamento por conta do tempo de execução. Esses resultados são apresentados nas Tabelas 5.11 e 5.12.

Tabela 5.11: Resultados para a representação *word2vec* com remoção de *outliers* para as bases de dados IMDB, Elec e Yelp com validação cruzada 5x2-*folds*. Células em destaque indicam os melhores resultados obtidos em termos de taxa de acerto. Legenda: acc (%): taxa de acerto; *stdev*: desvio padrão; Tempo (s): tempo total de treinamento e validação da CNN em segundos.

	IMDB			**Elec**			**Yelp**		
δ_{max}	**acc** (%)	**stdev**	**Tempo** (s)	**acc** (%)	**stdev**	**Tempo** (s)	**acc** (%)	**stdev**	**Tempo** (s)
-	90,19	0,235	6814,03	90,57	0,117	13232,45	92,97	0,163	3935,35
2.5	89,81	0,221	5202,63	90,22	0,113	10335,25	92,09	0,177	3010,48
2.6	90,1	0,206	5252,82	90,62	0,094	10368,34	92,65	0,099	3285,15
2.7	90,24	0,205	5371,27	90,73	0,177	11045,74	92,81	0,111	3390,52
2.8	90,3	0,205	5499,62	90,74	0,152	11194,92	92,95	0,096	3544,06
2.9	90,34	0,164	5602,07	90,75	0,182	11423,02	93,02	0,089	3622,86
3.0	90,39	0,229	6130,93	90,77	0,143	11631,30	93,06	0,074	3692,82
3.1	90,46	0,211	6230,39	90,8	0,086	11772,79	93,11	0,115	3749,5
3.2	**90,47**	**0,195**	**6275,3**	**90,84**	**0,173**	**11979,98**	**93,12**	**0,123**	**3799,15**
3.3	90,45	0,186	6354,6	90,79	0,116	12240,61	93,1	0,138	3839,45
3.4	90,44	0,182	6396,56	90,78	0,136	12463,43	93,08	0,15	3873,34
3.5	90,36	0,204	6355,92	90,73	0,205	12802,25	93,01	0,092	3889,43

Tabela 5.12: Resultados para a representação *gloVe* com remoção de *outliers* para as bases de dados IMDB, Elec e Yelp com validação cruzada 5x2-*folds*. Células em destaque indicam os melhores resultados obtidos em termos de taxa de acerto. Legenda: acc (%): taxa de acerto; *stdev*: desvio padrão; Tempo (s): tempo total de treinamento e validação da CNN em segundos.

	IMDB			**Elec**			**Yelp**		
δ_{max}	**acc** (%)	**stdev**	**Tempo** (s)	**acc** (%)	**stdev**	**Tempo** (s)	**acc** (%)	**stdev**	**Tempo** (s)
-	87,01	0,389	1838,29	88,45	0,346	3526,13	90,53	0,245	989,55
3.0	87,27	0,208	1751,7	88,67	0,23	3293,09	**90,8**	**0,223**	**958,77**
3.1	87,22	0,194	1768,55	88,67	0,174	3290,72	90,79	0,229	957,38
3.2	**87,31**	**0,143**	**1754,5**	**88,68**	**0,229**	**3298,66**	90,77	0,223	959,47
3.3	87,27	0,188	1767,97	88,66	0,231	3295,44	90,77	0,226	959,73
3.4	87,25	0,185	1763,51	88,63	0,219	3302,29	90,77	0,213	965,1
3.5	87,2	0,234	1766,38	88,61	0,176	3307,23	90,76	0,239	967,01

Observando esses resultados, verifica-se que para o *word2vec*, os melhores resultados foram obtidos com limiares maiores do que os resultados com as bases de testes pré-definidas. No entanto, observa-se também que a taxa de acerto da remoção de *outliers* ultrapassa a taxa de acerto sem a remoção a partir dos mesmo limiares. Para o *gloVe*, os resultados entre os limiares

de δ_{max} são muito próximos, uma vez que a quantidade de *outliers* removidos entre os limiares é pequeno. Similar aos resultados obtidos com as bases de testes pré-definidas, a taxa de acerto foi maior com a remoção de *outliers* para todos os limiares de δ_{max} para as três bases de dados. Também observa-se a queda no tempo de execução, que aumenta conforme o limiar utilizado.

Considerando os resultados dos experimentos apresentados nas Tabelas 5.7, 5.9, 5.11 e 5.12, foram conduzidos testes-T para verificar a probabilidade de significância dos resultados. Esse teste é utilizado para cálculo de valor-p com o intuito de rejeitar uma hipótese nula baseado em um limiar de nível de significância α. Valor-p é a probabilidade, sob uma hipótese nula, de se observar um valor da estatística de teste igual ou maior do que o encontrado, sendo que quanto menor o valor-p, maior a significância, pois a hipótese nula é rejeitada baseada no limiar α (Johnson, 1999). Nesses experimentos, a hipótese nula é de que não há diferença entre os resultados da CNN com e sem remoção de *outliers*. O limiar de nível de significância foi definido como $\alpha \leq 0,05$, ou seja, sobre a hipótese nula, espera-se que em menos de 5% das vezes um valor seja maior ou igual à estatística de teste. Os valores-*p* dos resultados são apresentados nas Tabelas 5.13 e 5.14.

Tabela 5.13: Valores-p dos experimentos com validação cruzada 10x10-*folds* e 5x2-*folds* apresentados nas Tabelas 5.7 e 5.11 para a representação *word2vec*. Células em destaque indicam valores acima do valor de α estabelecido (0,05).

	MR	**Subj**	**IMDB**	**Elec**	**Yelp**
δ_{max}	***valor-p***	***valor-p***	***valor-p***	***valor-p***	***valor-p***
2.5	$1,2 \cdot 10^{-7}$	$1,1 \cdot 10^{-6}$	$8,1 \cdot 10^{-6}$	$9,1 \cdot 10^{-6}$	$2,5 \cdot 10^{-09}$
2.6	$7,2 \cdot 10^{-6}$	0,00005	**0,08371**	**0,19272**	$1,2 \cdot 10^{-05}$
2.7	0,00075	0,00046	**0,16734**	0,01219	0,0002
2.8	0,01425	**0,09309**	0,04891	0,00289	**0,3433**
2.9	**0,06228**	**0,08747**	0,01564	0,01261	**0,20319**
3.0	0,01233	0,00935	0,00164	0,00215	0,03242
3.1	$6,2 \cdot 10^{-9}$	$2,9 \cdot 10^{-7}$	$9,2 \cdot 10^{-06}$	0,00008	0,00197
3.2	$5,5 \cdot 10^{-7}$	$9,2 \cdot 10^{-8}$	0,00053	0,00065	0,0006
3.3	$5,6 \cdot 10^{-6}$	$9,3 \cdot 10^{-5}$	0,00205	0,00048	$3,2 \cdot 10^{-05}$
3.4	$1,6 \cdot 10^{-5}$	$7,1 \cdot 10^{-6}$	0,0047	0,00106	0,00238
3.5	**0,17175**	**0,25183**	0,00288	0,01172	**0,17488**

Tabela 5.14: Valores-p dos experimentos com validação cruzada 10x10-*folds* e 5x2-*folds* apresentados nas Tabelas 5.9 e 5.12 para a representação *gloVe*. Células em destaque indicam valores acima do valor de α estabelecido (0,05).

	MR	**Subj**	**IMDB**	**Elec**	**Yelp**
δ_{max}	***valor-p***	***valor-p***	***valor-p***	***valor-p***	***valor-p***
3.0	$3,6 \cdot 10^{-5}$	0,00792	0,00745	0,01331	0,00062
3.1	0,0297	0,00196	0,02962	0,01591	0,0022
3.2	$7,9 \cdot 10^{-8}$	0,00668	0,00619	0,00943	0,006
3.3	0,00024	0,00082	0,0197	0,00778	0,00443
3.4	0,02037	0,00126	0,01668	0,03514	0,00171
3.5	0,00202	0,00178	**0,05006**	**0,05606**	0,00488

Considerando os valores-p apresentados na Tabela 5.13, observa-se que para as cinco bases de dados, a hipótese nula é rejeitada para a maioria dos limiares, com exceção de alguns valores intermediários e para o limiar δ_{max} = 3,5 exceto para a base IMDB. Para esses valores limiares intermediários em que a hipótese nula não foi rejeitada, é possível verificar que a taxa de acerto é muito próxima à taxa de acerto da CNN sem a remoção de *outliers*.

Considerando a Tabela 5.14, a hipótese nula pode ser rejeitada para todos os limiares considerando as bases de dados MR, Subj e Yelp. Para as bases de dados IMDB e Elec, com $\delta = 3,5$, os valores-p excederam o limiar de significância, o que sugere que a partir desse ponto a remoção de *outliers* tem pouco impacto na taxa de acerto da CNN.

5.2.2 Análise de *Outliers*

Para verificar quais palavras são consideradas como *outliers* pelo algoritmo de agrupamento baseado em densidade, considerando os *outliers* dos limiares que obtiveram os melhores resultados com as validações cruzadas 10x10-*folds* e 5x2-*folds* apresentados nas Tabelas 5.7, 5.9, 5.11 e 5.12, foi utilizada a biblioteca NLTK[7] (*Natural Language Toolkit*) (Bird et al., 2009) para verificar a quais classes gramaticais pertencem essas palavras. As classes gramaticais dos *outliers* em porcentagem, são apresentadas nas Tabelas 5.15 e 5.16 para as representações *word2vec* e *gloVe*, respectivamente.

Tabela 5.15: Porcentagem das classes gramaticais dos *outliers* obtidos com os melhores limiares de ρ_{min} e δ_{max} para as cinco bases de dados com a representação *word2vec*. Foram considerados os melhores resultados em termos de taxa de acerto obtidos com validação cruzada.

Classe gramatical	**MR**	**Subj**	**IMDB**	**Elec**	**Yelp**
Substantivo	81,75	84,01	82,14	81,4	82,61
Verbo	10,79	10,23	12,06	11,96	11,96
Adjetivo	6,12	4,72	4,77	5,45	4,37
Advérbios	1,29	0,86	0,90	0,87	0,87
Numeral	0,05	0,18	0,13	0,32	0,17
Modais	-	-	-	-	0,02

Tabela 5.16: Porcentagem das classes gramaticais dos *outliers* obtidos com os melhores limiares de ρ_{min} e δ_{max} para as cinco bases de dados com a representação *gloVe*. Foram considerados os melhores resultados em termos de taxa de acerto obtidos com validação cruzada.

Classe gramatical	**MR**	**Subj**	**IMDB**	**Elec**	**Yelp**
Substantivo	86,24	88,47	89,60	87,66	89,31
Verbo	5,83	4,23	4,22	4,41	4,47
Adjetivo	6,06	5,58	4,43	4,85	4,18
Advérbios	1,58	1,24	1,27	1,19	1,14
Numeral	0,29	0,48	0,48	-	0,90
Modais	-	-	-	1,89	-

[7]https://www.nltk.org/

Observando esses dados, verifica-se que para as duas representações e todas as bases de dados, a grande maioria dos *outliers* pertencem à classe substantivo, sendo poucos os *outliers* pertencentes às demais classes. Por conta da caracterização do problema de análise de sentimento, em que a polaridade dos documentos pode ser definida por algumas palavras chaves, justifica-se a remoção de uma grande quantidade de palavras sem uma perda substancial em termos de taxa de acerto, visto que a maior parte dos *outliers* são substantivos, e essa classe gramatical não caracteriza sentimentos e opiniões como as classes adjetivo e verbo.

Além dessas questões, vale ressaltar a natureza das representações pré-treinadas utilizadas, que não foram treinadas com corpus específicos de análise de sentimento. Dessa maneira, é possível que representações treinadas especificamente para o problema de análise de sentimento sejam capazes de capturar melhor a relação entre palavras para esse contexto, e consequentemente resultar em um melhor desempenho na aplicação do algoritmo de agrupamento baseado em densidade.

5.3 SÍNTESE

Analisando os resultados apresentados anteriormente, pode-se observar que a remoção de *outliers* teve impacto no desempenho da CNN, tanto em termos de taxa de acerto quanto em tempo de execução: a taxa de acerto melhorou ligeiramente para as duas representações sobre todas as bases de dados, e o tempo de execução baixou para as duas representações, de maneira mais significativa para o *word2vec*, visto que a dimensão dos vetores de palavras é seis vezes maior do que os vetores de palavras do *gloVe*.

Os experimentos para avaliação de significância estatística sugerem que embora o impacto na taxa de acerto da rede tenha sido pequena, a remoção de *outliers* tem influência no desempenho do classificador. Dessa maneira, a escolha dos limiares de ρ_{min} e δ_{max} do algoritmo de agrupamento pode ser um *trade-off* entre taxa de acerto e tempo de execução.

Embora o aumento na taxa de acerto tenha sido pequeno, foi possível remover grande parte do vocabulário das bases de dados sem obter uma queda significativa na taxa de acerto, o que sugere que muitas palavras de um documento têm pouco impacto na definição da polaridade do mesmo. Como consequência da remoção de grande parte do vocabulário, o impacto no tempo de execução é maior, uma vez que as representações das palavras são consideradas como parâmetros da CNN na arquitetura utilizada. Também vale ressaltar que, considerando os resultados com validação cruzada, os limiares ideais de δ_{max} são próximos para as duas representações, variando entre $\delta_{max} = 3$ e $\delta_{max} = 3,2$.

O baixo ganho na taxa de acerto pode ser justificado pelas representações pré-treinadas utilizadas, pois as representações foram treinadas com corpus de outros contextos: o *word2vec* foi treinado com o corpus de notícias *Google News* e o *gloVe* foi treinado com os corpus *Wikipedia2014* e *Gigaword5*. Como observado em trabalhos relacionados, o uso de representações treinadas especificamente para o contexto de análise de sentimento resulta em maior impacto na taxa de acerto (Severyn e Moschitti, 2015): isso sugere que a aplicação do modelo proposto sobre representações treinadas especificamente para análise de sentimento também pode resultar em um maior impacto na taxa de acerto.

6 CONSIDERAÇÕES FINAIS

Devido ao grande avanço em hardwares e o crescimento da Internet, uma grande quantidade de dados e informações são disponibilizadas com fácil acesso. Por conta desse grande volume de informações disponíveis, muitas áreas de pesquisa importantes ganharam renovado interesse e tornaram-se bastante ativas, dentre elas a área de análise de sentimento. Análise de sentimento é um importante problema da área de Processamento de Linguagem Natural, que estuda as opiniões, sentimentos, atitudes, avaliações e emoções de pessoas em relação a um determinado assunto, produto, serviço, entre outros. Por conta das suas diversas aplicações, como serviços financeiros, marketing e serviços a clientes, são necessários meios de automatizar essa tarefa devido à grande quantidade de informação.

Análise de sentimento pode ser abordada como um problema de classificação de texto, que consiste em automaticamente atribuir documentos de texto a classes previamente definidas. Para resolução dessa tarefa, podem ser utilizadas técnicas de aprendizado de máquina. No entanto, para que possam atingir uma boa capacidade de generalização, essas técnicas dependem de um pré-processamento cuidadoso e de uma representação adequada dos dados.

Este trabalho propôs tratar essas questões fundamentais por meio da aplicação de algoritmos de agrupamento baseado em densidade e redes neurais convolucionais. As representações de palavras utilizadas, conhecidas como *word embeddings*, são treinadas com relação ao seu contexto, capturando informações sintáticas e semânticas das palavras, o que faz com que palavras similares sejam projetadas próximas no espaço semântico.

Dessa maneira, o modelo proposto utiliza um algoritmo de agrupamento no espaço semântico para extrair informações adicionais das representações vetoriais das palavras com o objetivo de melhorar o desempenho da rede neural convolucional. Utilizou-se um algoritmo de agrupamento baseado em densidade para detecção e remoção de *outliers* dos documentos, antes desses documentos serem treinados e classificados pela rede neural convolucional.

Para análise do modelo proposto, foram conduzidos experimentos com dois algoritmos de obtenção de *word embeddings* sobre cinco bases de dados, estudando-se o impacto da remoção de *outliers* na rede neural convolucional. Para isso, foram executados experimentos para determinar regiões de baixa densidade utilizando o algoritmo de agrupamento com diversas variações de parâmetros. Com a definição dessas regiões, as palavras pertencentes a essas regiões são consideradas como *outliers* e removidas dos documentos que são treinados e avaliados pela rede.

Os resultados demonstraram que os *outliers* têm pouco impacto na taxa de acerto do classificador, podendo aumentar ligeiramente com a remoção de alguns *outliers*, mas também sendo possível remover muitos *outliers* dos documentos obtendo uma queda pouco significativa na taxa de acerto. Por outro lado, como o modelo proposto utiliza as representações como parâmetros da rede neural convolucional com o intuito de aprimorar essas representações, a remoção de *outliers* dos vocabulários impacta positivamente no desempenho em termos de tempo de execução da rede, uma vez que o número de parâmetros a serem treinados é menor.

A abordagem proposta apresenta limitações que podem proporcionar trabalhos futuros. Dentre elas, vale destacar:

- As operações de pré-processamento e detecção de *outliers* foram implementadas de maneira sequencial, enquanto as operações da CNN executam em paralelo na GPU. Como consequência, os resultados referentes a tempo de execução consideram somente o tempo de treinamento da rede;

- Os *outliers* são detectados e eliminados dos documentos antes do treinamento da CNN. Como as representações de palavras são treinadas como parâmetros da rede, a projeção das palavras no espaço semântico muda de acordo com cada época do processo de treinamento, no entanto, não são detectados novos *outliers* baseados nessas novas representações;

- Foram conduzidos experimentos considerando somente uma métrica de distância/similaridade: distância Euclidiana. Experimentos com outras métricas, como correlação de Pearson e similaridade de cossenos, e análises da influência da medida de distância podem impactar no desempenho do modelo e na análise dos resultados.

Outras propostas de trabalhos futuros incluem: execução de novos experimentos em outras bases de dados para comparação com outros algoritmos, adaptação dinâmica das regiões de baixa densidade com o intuito de detectar e descobrir novos *outliers* em ambientes online, aplicação do modelo proposto usando representações pré-treinadas especificamente para o problema de análise de sentimento, modificações na arquitetura da rede e o uso de diferentes algoritmos para detecção de *outliers* no espaço semântico.

REFERÊNCIAS

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. e Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. et al. (2016). Tensorflow: A system for large-scale machine learning. Em *OSDI*, volume 16, páginas 265–283.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. *CoRR*, abs/1206.5533.

Bird, S., Klein, E. e Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. "O'Reilly Media, Inc.".

Bouckaert, R. R. (2003). Choosing between two learning algorithms based on calibrated tests. Em *ICML*, volume 3, páginas 51–58.

Cambria, E. e White, B. (2014). Jumping nlp curves: a review of natural language processing research [review article]. *IEEE Computational Intelligence Magazine*, 9(2):48–57.

Chowdhury, G. G. (2003). Natural language processing. *Annual review of information science and technology*, 37(1):51–89.

Collobert, R. e Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. Em *Proceedings of the 25th international conference on Machine learning*, páginas 160–167. ACM.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K. e Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Conneau, A., Schwenk, H., Barrault, L. e Lecun, Y. (2016). Very deep convolutional networks for natural language processing. *arXiv preprint arXiv:1606.01781*.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation*, 10(7):1895–1923.

Dos Santos, C. N. e Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. Em *COLING*, páginas 69–78.

Duda, R. O., Hart, P. E. e Stork, D. G. (2001). *Pattern classification*. John Wiley & Sons.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X. et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. Em *KDD*, volume 96, páginas 226–231.

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, 3(Mar):1289–1305.

Go, A., Bhayani, R. e Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12).

Goldberg, Y. (2016). A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420.

Goodfellow, I., Bengio, Y. e Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Haykin, S. (1999). *Neural Networks, a comprehensive foundation*. Prentice Hall.

He, Z., Xu, X. e Deng, S. (2003). Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9):1641–1650.

Hinneburg, A., Keim, D. A. et al. (1998). An efficient approach to clustering in large multimedia databases with noise. Em *KDD*, volume 98, páginas 58–65.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N. et al. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Hodge, V. e Austin, J. (2004). A survey of outlier detection methodologies. *Artificial intelligence review*, 22(2):85–126.

Jain, A. K. e Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

Johnson, D. H. (1999). The insignificance of statistical significance testing. *The journal of wildlife management*, páginas 763–772.

Johnson, R. e Zhang, T. (2015a). Effective use of word order for text categorization with convolutional neural networks. Em *NAACL-HLT*, páginas 103–112.

Johnson, R. e Zhang, T. (2015b). Semi-supervised convolutional neural networks for text categorization via region embedding. Em *Advances in neural information processing systems*, páginas 919–927.

Kalchbrenner, N., Grefenstette, E. e Blunsom, P. (2014). A convolutional neural network for modelling sentences. Em *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, páginas 655–665, Baltimore, Maryland.

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kingma, D. P. e Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Kohavi, R. et al. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. Em *Ijcai*, volume 14, páginas 1137–1145. Montreal, Canada.

Krizhevsky, A., Sutskever, I. e Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Em *Advances in neural information processing systems*, páginas 1097–1105.

LeCun, Y., Bengio, Y. e Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. e Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

LeCun, Y., Bottou, L., Bengio, Y. e Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Liddy, E. D. (2001). Natural language processing. Em *Encyclopedia of Library and Information Science*. Marcel Decker, Inc., 2 edition.

Liu, B. (2012). *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y. e Potts, C. (2011). Learning word vectors for sentiment analysis. Em *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, páginas 142–150. Association for Computational Linguistics.

Manning, C. e Schutze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press.

McAuley, J. e Leskovec, J. (2013). Hidden factors and hidden topics: understanding rating dimensions with review text. Em *Proceedings of the 7th ACM conference on Recommender systems*, páginas 165–172. ACM.

Mikolov, T., Chen, K., Corrado, G. e Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. e Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. Em *Advances in neural information processing systems*, páginas 3111–3119.

Pang, B. e Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. Em *Proceedings of the 43rd annual meeting on association for computational linguistics*, páginas 115–124. Association for Computational Linguistics.

Pang, B. e Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. e Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pennington, J., Socher, R. e Manning, C. D. (2014). Glove: Global vectors for word representation. Em *EMNLP*, volume 14, páginas 1532–1543.

Rodriguez, A. e Laio, A. (2014). Clustering by fast search and find of density peaks. *Science*, 344(6191):1492–1496.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747.

Rumelhart, D. E., Hinton, G. E. e Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

Sebastiani, F. (2002). Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47.

Severyn, A. e Moschitti, A. (2015). Twitter sentiment analysis with deep convolutional neural networks. Em *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, páginas 959–962. ACM.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A. e Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. Em *EMNLP*, volume 1631, páginas 1631–1642.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I. e Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

Turian, J., Ratinov, L. e Bengio, Y. (2010). Word representations: a simple and general method for semi-supervised learning. Em *Proceedings of the 48th annual meeting of the association for computational linguistics*, páginas 384–394. Association for Computational Linguistics.

Wang, P., Xu, J., Xu, B., Liu, C.-L., Zhang, H., Wang, F. e Hao, H. (2015). Semantic clustering and convolutional neural network for short text categorization. Em *ACL (2)*, páginas 352–357.

Wiwie, C., Baumbach, J. e Röttger, R. (2015). Comparing the performance of biomedical clustering methods. *Nature methods*, 12(11):1033–1038.

Xu, R. e Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3):645–678.

Zeiler, M. D. e Fergus, R. (2013). Visualizing and understanding convolutional networks. *arXiv preprint arXiv:1311.2901*.

Zhang, X., Zhao, J. e LeCun, Y. (2015). Character-level convolutional networks for text classification. Em *Advances in neural information processing systems*, páginas 649–657.

Zhang, Y. e Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.